A multi-family multi-processor education and development system.

Whitworth, P. F.

Award date:
1983

Awarding institution:
University of Bath

A MULTI-FAMILY MULTI-MICROPROCESSOR EDUCATION AND DEVELOPMENT SYSTEM

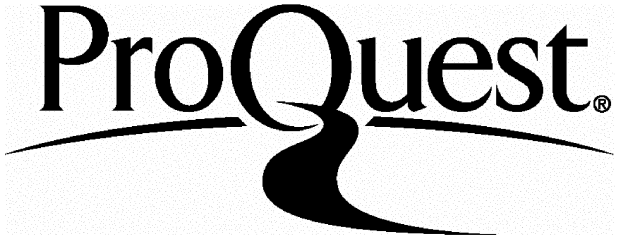
submitted by P. F. Whitworth

for the degree of Ph. D

of the University of Bath

1983

COPYRIGHT



To

Jim, my Father

Joan, my Mother

and Margaret, my Wife

Contents

SUMMARY

The rapid development of the microprocessor has not been matched by the development of equipment to assist in the education of those who must use the devices. As a result several different methods, based upon equipment designed for other tasks, have become predominant.

This dissertation examines each of the commonly employed approaches and compares their advantages and disadvantages. From this examination, a specification for a system designed primarily to support microprocessor users through both the stages of their own development, and those of the equipment they design. The implementation of a system based upon this specification (the main feature of which is the use of several microprocessors to assist the user) is shown for several common microprocessor families.

The thesis contains both an appraisal of the future viability of the concept and a bibliography of recent papers in the field of microprocessor education.

Abbreviations

This section lists the principal abbreviations used. Others (such as the names of signals) are given on their first occurence in the text.

CPU	Central Processing Unit
DMA	Direct Memory Access
ECL	Emitter Coupled Logic
EPROM	Eraseable Programmable Read Only Memory
IC	Integrated Circuit
IO	Input/Output
LED	Light Emitting Diode
LSI	Large Scale Integration
LSTTL	Low Power Schottky Transistor Transistor Logic
RAM	Random Access Memory
ROM	Read Only Memory
TTL	Transistor Transistor Logic
VDU	Visual Display Unit

1. Introduction

The field of electronics is widely regarded as being the fastest moving of the many facets of human endeavour. It is also generally accepted that the fastest developing area within the field of electronics is that associated with the development of the integrated circuit computer: the microprocessor.

The explosive growth of this field dates from 1971 with the introduction of the '4004' by the Intel Corporation[1]. This device was an attempt to design the 'once and for all' calculator chip. Until this time, each new facility provided for a calculator required a total redesign of the integrated circuits that gave the calculator its properties. The idea that Intel implemented in the 4004 was that, if the basic capabilities required for any calculator were available in a machine so that they could be executed in any desired sequence by placing their identifier in a list, the facilities provided by the calculator could be radically altered merely by changing the list.

This is, of course, the principle of the stored program computer which was already a proven and established tool. The difference was in the number of integrated circuits required to implement an entire machine. This difference influences size and price. In the event, Busicom, the Japanese company that commissioned Intel to design the calculator IC went bankrupt, leaving Intel to discover the ready market that appeared for the new device.

In the intervening years, the power of the devices produced has increased, to such an extent that the latest microprocessor devices deliver power comparable with that of mainframe computers of the mid 1960's. In conjunction with the increased power of the processors has come a similar increase in the power of the support chips. For example memory chips hold many times the information and can retrieve it faster, whilst using less power and space. Input/Output devices, originally confined to the role of voltage/current translators, are now highly intelligent in their own right and

provide significant support for the processor.

The race shows no sign of slowing and whilst this leads to enormous leaps in technological achievement, it has also introduced major problems for those who must learn, or educate people in these topics. Similar problems occur when people who have finished their formal education have to develop, test and support electronic systems based upon increasingly more recent, powerful, complex and cost effective integrated circuits.

The aim of this work is to critically examine the currently used techniques, and in the light of results obtained from this examination, devise a possible new solution to the problems.

Before any microprocessor based unit can be put into full production (which is assumed to be the ultimate goal of any programme of education and application), there are several distinct phases of development that must be passed through, not all of which are electronic development stages. The stages can broadly be classified as a general awareness of the available technology and the application of that technology to a particular task.

Each engineer who works on microprocessor based equipment has to go through a learning process to enable him to develop the skills necessary to successfully design, build and debug a microprocessor based system. Some skills will be an extension of older electronic skills whilst others will be totally new. The learning process may have occurred within the context of a University/Polytechnic degree course in electronic engineering or computer science, or it may have taken place after this structured, formal phase of education has finished and would then be by a 'refresher' or 'up-date' course, or by self education. It is probable that not all available devices will have been covered, or all possible approaches to problem solution examined. Even if the course had covered every currently available device, new versions, or completely new devices, will have appeared in the interval between the end of

the course and the time that the knowledge will be applied. It follows that the engineer will have to up-date his knowledge at regular intervals.

The company, as a corporate entity, must also obtain knowledge of microprocessor techniques. The knowledge required ranges from the difficulty of identifying programmed read only memories that are held in company stores (there may be four or five different programmed versions, with no obvious differences between the integrated circuits) through the problem of how to depict/specify a computer program (with the inevitable revisions) for drawing office purposes, to a general awareness of the strengths and weaknesses of a microcomputer based product which can be tested electrically long before the product is specified, yet can never be proved 100% working at any time. The information required for company purposes varies in accordance with the task of the individual concerned and really amounts to an awareness campaign. Once this stage has been passed the appropriate practices will become normal company policy and will survive the inevitable changes of personnel.

At some later stage the engineer will have to examine a range of microprocessors *with a particular application in mind*. This will take place when the company is developing its first microprocessor based product or when the processor in use until then has proved incapable of the new task or has become obsolete. Usually organizational pressures will force the adoption of one microprocessor for as long as possible as the reduction in research equipment, company expertise, stock levels and field service replacement stocks are some of the major attractions of a microprocessor based system. The fact that a given application is to be implemented will seriously affect the relative desirability of the different microprocessors.

The final stage of development is that associated with the production of a working, testable hardware/software combination that performs the required functions. This phase will occur at least once for each project undertaken and

may represent a time period of several man weeks to many man years. This phase is likely to be the only one which receives serious attention in terms of budget allocation.

Each of the above mentioned stages represents a problem of education associated with the same techniques and topic, yet the demands, problems and finance available at each stage differ radically. The net effect is that each of the phases has forced the production of a different solution and no coherent approach that enables the entire spectrum of microprocessor related education to be taught with continuous support from one tool has emerged.

The other major use of microcomputers has been the provision of desk-top computing facilities which concentrate entirely on the teaching, production and use of software and programming techniques. Such facilities are desirable because they are cheap, available on demand and are under the direct control of the user. In terms of software education, the problems encountered with microprocessor based equipment are not significantly different from those that have been met with teaching based upon minicomputers or mainframe computers. In each case the architecture of the machine is hidden behind a wall of standard programming languages (ALGOL, FORTRAN, C, PASCAL, BASIC) and operating systems (UNIX, CP/M). It is therefore possible for a user to be unaware as to the type of machine he is using.

So, whilst it is desirable that any unit used as part of these general education and development stages, lends itself to the teaching of software, this problem is a lot less severe. Most of the discussions will centre on the teaching of hardware on the assumption that any unit capable of this task will be capable of use as a system software teaching aid.

2. Educational Requirements

The requirements at each level of development differ widely. At the first stage, when the student initially encounters the many concepts involved in microprocessor topics, any practical work should be designed to assist in the assimilation of these concepts. Exercises at this stage should be highly structured and supportive and the student should be led towards the correct understanding of a particular concept by attaining a sequence of intermediate goals, with each successful completion indicating the way forward. The concept of 'selective failure' is significant. The student should not be thwarted in pursuit of a goal by an incorrect application of manipulative tools, yet must not be allowed to attain a goal by the incorrect application of a concept. An obvious example is the use of program execution commands where the letter 'E' is frequently used alone to indicate that the program should start from its current state, whilst 'E1000' would indicate that the program at location 1000 should be started. As all these addresses are in hexadecimal, 'EE' is valid but is usually obtained as a consequence of miskeying, rather than a desire to run whatever program might start at location 'E'. It is inevitable that the program residing at 'E' will destroy the recent programming efforts of the student. Again, a paradox becomes apparent when the student must learn about the correct use of manipulative commands since they are necessary parts of any computer system, yet, until the concepts are completely understood, the objects being manipulated, and thus the tools, are meaningless.

Resolution of this paradox can be achieved if the tools are automatic in operation. In a sense, all computing related topics must be computer assisted. The major points of interest within a computer system are usually the values of the internal processor registers, the states of the input/output devices and memory locations currently holding program, data variables and stack data. Correct display of these elements will indicate such things as interrupt

status and subroutine nesting depth. Access to these locations is usually by register, input/Output and memory examination commands but experience shows that such commands interfere greatly with the learning process by requiring the use of sequences, whose logic is not yet understood. This difficulty is compounded by the excessive demands of finding the appropriate letters on the keyboard at this very early stage. The provision of automatic display facilities leads to an important benefit. Just as chess playing computers can be excellent tutors by demonstrating the wide choice of moves available, so the student of microprocessors benefits from a display system that indicates all the side-effects of an action rather than leaving the student to examine only the effects expected. Another crucial feature required of display facilities is data translation since, at this stage, the student will not be capable of the automatic recognition of number patterns as being machine code, text, data, stack data or meaningless rubbish. It is important that the data be offered in as many forms as possible in order that such recognition can be achieved.

Any hardware associated with a practical investigation of topics in microprocessing should *appear* simple at the level at which it is being examined. Printed circuit boards should contain as few components as possible, consistent with providing a machine capable of useful work, preferably one of each type of device needed. The immediate location and recognition of the components discussed in lectures will assist in making the equipment less mysterious. If the boards are overly large and complex, they will not be identified with the simpler systems met theoretically. Of course, providing the boards appear simple, the circuit can be as complex as required. This can be achieved either by the use of programmable devices in place of larger quantities of non-programmable parts, or by the removal of any additional circuitry onto additional concealed cards.

The final requirement of the initial contact stage is an ability to demonstrate which features are constant over several microprocessor families and which are apt to change. If this is to be successfully achieved, the removal of incidental changes is of the utmost importance. For example, if the student is told to examine two microcomputer systems by different manufacturers which use different keyboard commands to examine memory (perhaps E(xamine) and M(emory)), this will be the first difference encountered and will tend to obscure the essential differences such as numbers of registers and addressing modes.

After the concepts are thoroughly understood, the student is likely to encounter other microprocessor families. Each of these families will be compared and contrasted with those already encountered to determine their relative strengths and weaknesses. At this stage, the ability to examine the devices using a common set of manipulative tools will be the most important feature of any practical work. The previously desirable, highly supportive system will now be required to allow the intelligent short cuts that reduce the time taken to quantify the new device and its support family. Display facilities will be required to reduce the information presented to more compact forms to allow more rapid examination of the data. Typically, data will only be required in instruction form (where applicable) and hexadecimal form if data is being displayed.

Practical work is likely to concentrate on the areas of instruction set and hardware details in which the processor appears to differ from those previously encountered. As students will have examined processors in different orders, it is impossible to say which particular details will strike them as different. At this stage, not all the implications of each feature will be meaningful, they will merely be novel. Because of this, it is likely that not all unexpected side-effects will be examined further but in practice there are

some side-effects that are never observed.

It is only when a particular application is under consideration that a full evaluation of several processors can be undertaken and this represents the third developmental stage. As soon as the constraints of a practical problem are applied, several microprocessor families will be discarded on the basis of knowledge gained during examination of that family for update purposes. It is generally at this stage that the dangers of introducing students to only one processor family become apparent. Instead of checking on available devices to locate the most cost effective processor which can support the product line for many years to come, students exposed to only one processor will check whether that processor can cope. If that is the case, the processor will be used regardless of its suitability. There are many cases where companies have decided on a processor family not because of its merits, but because it was the only one to which their first microprocessor experienced engineer had been exposed. If the engineer is familiar with several families, he will not only be able to choose from a wider selection initially, but will also possess the ability to examine new devices for suitability. Of particular importance is the ability to derive data direct from manufacturers' data sheets, a skill which is very necessary but rarely taught.

Once the choice has been narrowed to one or two particular devices (by whatever method), the next requirement is for a unit on which any crucial points of understanding can be investigated. Prior analysis of the problem is likely to produce several constraints that will be difficult to satisfy, either in terms of timing or repeatability. Unless the engineer is highly skilled, proof of the correct operation will be required by demonstration, rather than calculation. This is most easily achieved if a working system is already available, preferably capable of demonstrating all the features and

facilities of the particular device, yet offering a good level of support. The support will be required as the engineer, although competent with microprocessor concepts and indeed other microprocessors, will not necessarily have experience of the chosen device(s). From the project viewpoint, it is also desirable that a working prototype system be demonstrated as soon as possible. This will enable an examination by those responsible for commissioning the unit and allow the changes of specification that usually follow from a demonstration of a working product. This implies that the unit is initially built using a family of pre-built and tested modules wherever possible, so that only the software need be written at this stage. As software is often the hardest part of the project to specify and produce (certainly the most expensive part of the development), any aids are beneficial, particularly if two competing processor families are being demonstrated. It is therefore desirable that the software is initially written in a high level language, even though severe time and memory size penalties will be incurred. Compilers can then be used to transfer the task to the appropriate machine code. Usually the resultant system will be adequate for demonstration purposes and will assist in tightening the specification before more time is spent on production software.

The ready availability of a wide range of pre-built modules will allow for the correct positioning of the hardware/software trade off. It is usually true that tasks can be performed with simple hardware using complex software or vice-versa. As software is expensive in development but cheap in production, whilst hardware is the reverse, the correct balance between the two must be maintained. This is even more difficult where there are several mutually exclusive trade offs that can be manipulated. Such a case occurs if the system performs several, related complex tasks.

As the time and memory size constraints will have been abandoned to allow

the use of high level languages, the prototype is now likely to consist of a pre-built modular hardware, a 'conceptual software system' that demonstrates the major features of the system and a selection of the more crucial sections of software and hardware available to prove the ability of the system to perform the task at production level. It will now be possible to move forward to the final development stage, where one family is selected and a complete integrated production hardware/software unit is produced and debugged.

It is usually at this stage that sufficient finance is available to meet all the requirements of a design engineer. A microcomputer will have to be constructed that provides the required facilities, in terms of program space and input/output facilities, yet meets the constraints of power dissipation, board size and production cost. This unit must be made functional, and there are several techniques for achieving this. The simplest method is to use in-circuit emulation. Here the processor, which by the nature of the system has complete control of all other devices, is replaced by a cable leading to a complete test micro or minicomputer. The microcomputer provides facilities for examination of all registers, memory and input/output locations and is designed so that programs can be executed even in the presence of memory faults. This enables the exercising of the system under test. The problems of testing microcomputer systems are usually aggravated by the long time period of any waveforms involved. If the waveforms can be made simpler and more repetitive, they are amenable to study by oscilloscope rather than the far more expensive and complex logic analyzer. The simplest way of obtaining repeatable waveforms is by the use of short test programs. The running of these programs, which require a working memory/processor system, is simplified by memory simulation in the debug system. Software debugging is likely to revolve around the three facilities of single stepping the program so that a section of code which does not work can be examined in detail, data snapshots

in which the values of selected locations are automatically stored at intervals for later analysis and, finally, the ability to examine the sequence of events that leads up to some abnormal condition of the system. This is usually done by a trace facility in which each bus cycle is captured in a high speed memory, the capturing process stopping when the abnormal event is reached. The above three facilities will usually be arranged to operate under certain special conditions such as input/Output access only, instruction fetch only, only between locations XXXX and YYYY or combinations of the above.

Given the above desiderata, the next chapter will examine the popularly used techniques for providing microcomputer education and development and discuss their respective strengths and weaknesses.

3. Existing Systems

In the field of microprocessor development systems and education, four major classes of equipment have tended to predominate and the literature shows support for each of the various methods or combinations thereof. In this chapter, each of the common approaches will be discussed in terms of its suitability for use in the four developmental stages previously outlined. For each of the approaches a reference to representative paper will be given. Other relevant publications are listed in the bibliography, arranged according to the type of system advocated or discussed. In fact, the vast bulk of papers on microcomputer education over the past few years can be found in four issues of various journals[2,3,4,5].

When microprocessors were first introduced, the manufacturers realised that the radical differences between discrete logic implementation and a microprocessor based implementation of a task would, in itself, be a major factor governing the acceptance of the new devices. This led to the introduction of evaluation cards which are usually single board microcomputers. In the earliest models, a microprocessor would be supplied with a monitor program located in read only memory, a very limited amount of random access memory, one or two parallel input/output devices and a serial input/output device with a teletype interface. These cards were very difficult to expand since they had been designed to be as cheap as possible consistent with producing a demonstrable system. However it is in the best interests of the manufacturers to provide a high level of hardware and software support for their products and the preferred supportive tool is usually a development system produced by the manufacturer with typical prices of three to eight thousand pounds. Whilst such a sum is a reasonable capital cost to be amortized over a contract or product lifetime, it is unreasonable as part of a speculative development budget, where the low cost development cards are

widely used. The requirement for a teletype proved to be a limiting factor for many companies with no previous computer usage, as the development cards were in the region of one to three hundred pounds, whilst a teletype was three to six hundred pounds. The later versions recognised that many users would not possess terminals and so provided a hexadecimal keypad and display as an alternative to the teletype. Storage of programs was achieved using the paper tape reader/punch associated with the teletype or a domestic cassette recorder and modem circuit. Support for the user was generally limited to commands to examine registers and memory and alter them as required, insertion and removal of 'breakpoints', in which the program runs as normal until the specified location is reached, and the provision of program single stepping.

There can be little doubt that these boards spurred the development of the first hobbyist and home computers. However until this market was recognised, the microcomputer was seen primarily as an electronic circuit replacement, rather than a cheap, small computer. In turn, the development of the home computer led to the realization of the potential of the domestic television as a cheap display device. The latest generation of single board computers produced by the manufacturers have appeared in support of the new generation of sixteen bit microprocessors and, due to the complexity and cost of the processors they contain, some manufacturers have moved back to the visual display unit as being the only practicable display method. For example, the Motorola 68000 has sixteen, thirty-two bit registers to display and a hexadecimal display is not too convenient. Others, like the Intel 8086, have retained the hexadecimal keyboard and display. This places them at a serious disadvantage, since, with the introduction of improved architecture and reduced cost of all types of memory, it is now possible to offer such features as mnemonic assembly and symbolic debug as cost effective facilities. Both these facilities require that a full QWERTY keyboard and display be available.

In general, the manufacturers differ as to the extent of expandability made available to the purchaser of evaluation boards. Whilst most will allow the addition of extra standard memory or input/output devices, it is not uncommon for the more powerful control signals, such as interrupts, slow memory handshake etcetera, to be dedicated to providing facilities associated with the simple user interface that the units possess. Typically, the most important interrupt will be reserved for program abort or single step, while in many cases, full address decoding is not performed, thereby saving the cost of several integrated circuits.

The hobby computers have developed to a stage where there are many single board computers which are not produced by the manufacturers of a chip family and which can thus combine the best products of several manufacturers. In the main, the independently produced boards have used the Zilog Z80 and the MOS Technology (now owned by the largest independent board manufacturer, Commodore Business Machines) MCS6500 series, notably the 6502.

These facilities show a much wider divergence than the boards produced by manufacturers, largely because of the element of competition. For the manufacturers, the single board computer is a device to aid in the sale of chip families; for the independent manufacturers, it is the main business. It is therefore possible to buy single board computers with a full QWERTY keyboard, video interface and one high level language (usually BASIC) at a price below that of the manufacturers' minimal system.

Single board computers have frequently been used as the basis of a microcomputer teaching laboratory and a typical laboratory based on such units is described by Cahill, Laverty and McCokell[6]. The major advantage presented by the single board computers is one of cost. A typical first year electrical engineering intake of fifty to one hundred students will require a large number of workpoints if problems of timetabling are not to become acute. Ten

student positions can represent a large capital outlay. This leads to the attraction of boards costing a few hundred pounds, but, of course, such boards fall short of the ideal arrangement. When used during the initial stage of development, the boards suffer from several disadvantages. Due to the constraints imposed by their low cost, the boards that employ hexadecimal keyboards and displays cannot offer facilities such as editors and assemblers. In addition, it is likely that up to one third of the board area will be associated with keyboard and display functions. Those cards using a VDU interface will have simpler electronics but now require the addition of a visual display unit for each station with attendant costs. If such a board is used with the vdu, it will be able to support editors and assemblers only by increasing the size of read only and random access memory held on the board. This is due to the difference in size that exists between system software and typical small control tasks. For example, on the Z80, a small assembler will occupy about four kilobytes, a BASIC interpreter uses eight kilobytes, whilst a *major* control unit with a large number of embedded messages might also take eight kilobytes. Contrast this to the average single chip microcomputer with, at most, two kilobytes of read only memory or the typical final year university project software of several hundred bytes. If the system is designed to cope with the demands of such supportive software, it will inevitably be more complex. Further difficulties arise when the program must be run. As any editor/assembler must be both memory resident and edit/assemble from memory to memory (due to the lack of storage peripherals), it will be possible for the source code to be corrupted if the program is incorrect. This will mean extra time spent correcting or re-entering the programs.

As the cards are the products of different manufacturers, command formats will differ, presenting a serious handicap to the student at this stage. Problems will arise for those responsible for producing and maintaining

laboratories based on such units, if interfaces to equipment in the laboratory are required. This will occur because of the diverse bus standards used by the various manufacturers. For example, if a special laboratory interface was produced, the different bus timings , card and connector formats would necessitate much duplication of effort as each manufacturers card would require its own interface to the equipment. Such boards are also inadequate for system software education tasks, since, as stated, system software is typically far larger than the software for embedded computer controllers. At the up-date stage, which represents the market the cards are primarily aimed at, the differences in monitor commands will now represent an annoyance rather than a major handicap which will reduce in significance even further if the time intervals between updates on different processors is extended. As no major tasks are attempted at this stage, limitations of monitor and hardware will not prove to be too serious.

If a product prototype is being developed, the boards can form an invaluable starting point for the construction of dedicated hardware. The provision of a processor card that is known to be working provides the means for running exercise programs to assist in the testing of other sections of the hardware. As with the laboratory, there will be problems of interfacing if a company intends to build prototypes based on more than one manufacturers' product. The lack of support is likely to prove the major problem. Any project that requires more than a few lines of assembler code will suffer if an assembler is not used. Updating the software as errors are corrected, and providing adequate program documentation and description for debugging purposes become much more difficult. The use of an assembler requires an editor to construct and correct the input text. At the full hardware and software integration level, the prototype cards cannot offer any assistance other than an emergency supply of spare integrated circuits and test bed

facilities for suspect integrated circuits.

The independently manufactured single board computers offer better facilities, but this is usually achieved by the use of a high level language, typically BASIC, and thus the true architecture of the processor is hidden. This renders the board less useful as a teaching aid during the initial concept stage. The use of a full QWERTY keyboard and television set or monitor does improve facilities at low cost, but this is offset by the necessary increase in board complexity. At the up-date stage, the additional features, being rarely directed to assembly language support, render the cards little different to those supplied by the manufacturer. This is also true at prototype and full hardware/software integration levels.

The use of multiboard self-contained microcomputers also receives much support[7]. Such machines consist of a processor with large quantities of random access memory and software either in read only memory or on disk. The earliest systems were designed as hobbyist computers and cheap minicomputer replacements. The major handicap for the earliest units was the lack of fast, non-volatile back-up storage. It was not until an Intel employee, experimented with the addition of a floppy disk drive (designed by IBM to replace punch cards and paper tape as an original data entry medium), that the units could begin to compete with their larger brethren. At this time, the most widely used minicomputer was the PDP 8 series and an operating system was written for the Intel 8080 based on that used by DEC on the PDP-8. This was marketed as CP/M and, because of the uniform programming environment provided for many machines, software became available from many different sources. The uniform programming enviroment was achieved by taking the lowest common denominator in terms of hardware facilities, but, as microprocessors have developed, CP/M has been retained because of the large amount of software available. However, compatability must be retained if CP/M is to be useful and the changes made

are largely cosmetic. Several other companies developed products based upon the MCS6502 microprocessor, notably the PET and the Apple. Both these units were sold on the turnkey system approach, the user would not need to look inside the box. As no common 6502 operating system appeared, software had to be written especially for each of these machines, limiting the quantity available. One advantage enjoyed by the Apple over CP/M machines is that it possesses high resolution graphics as standard and Apple software writers were not slow to capitalize on this for games and graphics packages. It is an indication of the hold that CP/M has on the market that it is now possible to buy a plug-in Z80 sub-system for both the PET and Apple to enable them to run CP/M software.

More recently, using the latest generation sixteen bit processors, the new generation of self-contained microcomputers can simultaneously support several users, have in built twenty megabyte Winchester technology disk drives, megabyte memories and rival many minicomputers. Whilst they can still represent a cheap alternative to a minicomputer, the user does not have to lose any computing power.

All the self-contained microcomputers were designed primarily for use as software machines. Indeed, the most commonly used machines (Apples and S100 bus based) are both organized so that interference with the bus by outside devices for direct memory access purposes is difficult, if not impossible. When used at the initial concept stage, they can provide a good insight into the software activities of a microprocessor and if provided with a range of pre-built interfaces they offer a method of demonstrating interfacing software.

Apart from their higher capital cost, more than one type of machine will be needed if more than one processor is to be studied. As the debugging facilities for these machines tend to be more capable, the commands are more

numerous and complex and differences between the units will be a greater barrier. In addition, the problems associated with supporting these machines will become more severe. Transferring programs between machines that store the data in different formats, and the use of different suppliers and service agents all act to make this approach for teaching laboratories less attractive. At the up-date level, the units represent a large investment that is unlikely to be recoupable, unless the devices are then used as general purpose processors, when the objections mentioned above will appear. Prototype work can make use of such machines as a basis for adequate project software support, but the problem of incompatible input/output modules will again occur, as, despite the attempts of the Institute of Electrical and Electronic Engineers[8], no industry wide standard bus has emerged. Once again, the units have no hardware support to offer at the hardware/software integration stage.

The use of mainframe/mini-computer simulations is the third major approach taken to the problems of microcomputer education and development. In such systems, a program runs on a mainframe and provides an editor and assembler as well as simulating the action of a microprocessor system. The description here is of the MicroSim package by D. M. England & Partners Ltd. The user gains access to the computer as appropriate and using the normal commands of the system, the relevant simulator for the microprocessor of interest is executed. Once this occurs, the user is in a standard environment that changes little for all the computers the package is available for. The environment suffers from the requirement that the package be transportable to different computers in that the terminal handler is written in FORTRAN and only provides line editing capability. All lines are prefixed with a number which is used for all editing references and, as with the BASIC language, the lines of program will be executed in ascending line number order. Sections of program are created as segments and, whilst references to other objects is by simple

label, objects in other segments are referenced by composite labels of the form 'segmentname.labelname'. There are no assembly or link passes, so that, as each line is entered, it is compressed to an internal format based on the appropriate machine code. A request for a listing will produce a standard list format showing program counter, machine code, labels, source statements and comments. It is also possible to generate an output of machine code only, suitable for transfer to a microcomputer. Input/output is not available to real devices, and is either directed to the keyboard, where data can be entered in any base desired, or to a subroutine, written in the relevant assembler, which is called for each byte to be transferred. This is designed so that the user can simulate input/output devices of any complexity. There is currently no capability for input/output to files on the mainframe which limits the amount of processing that can be done on data gathered 'in the field'. An advantage of the simulator is that, if the program attempts to execute data or an instruction that conflicts with previously declared instruction boundaries, the simulator will report an error. This is a major benefit when students are first introduced to programming, as a very common error is to omit the program termination. As this feature can not be disabled, the ability to alter code during execution is not demonstrable, though some might argue that this is a positive constraint, rather than a negative one. Unfortunately, there is no program single step trace capability, and examination of the registers is somewhat awkward.

Due to the popularity of the CP/M operating system, Gilbert[9] at Surrey University has simulated not only a Z80, but also floppy disks and the routines to drive them. This package, written in FORTRAN and called HORACE (Horizon and CP/M Emulator) will run any CP/M program, and CP/M compatibility is ensured in that the first program loaded into the simulated Z80 is CP/M itself. A major feature of HORACE is that extremely powerful debug facilities

are provided. Whilst simple debug tasks can be requested easily, a debug language exists which can be used to create new debug commands that offer conditional execution, action on specific values of registers, or only between limited addresses.

The simulators offer many advantages. Where the time sharing computer is already available, as many student positions as necessary can access the simulator without difficulty. Due to the ability of the simulators to check the legality of an instruction before it is executed, many initial mistakes are caught and reported rather than destroying the entered program. Where the packages have been implemented on several different processors, students can simulate the same task on several machines, yet retain a common working environment with stable manipulative tools. The simulator can provide useful software support at all levels.

Theoretically, the most powerful approach to all phases of microprocessor education and development, but also the most expensive, is the use of in-circuit emulation based development systems. Designed by the manufacturers to be the preferred tool for prototype and full hardware/software integration, they operate on the basis that, as the microprocessor controls the system, control of the microprocessor provides all the necessary access to the system. To provide the appropriate degree of control, a microprocessor IC is surrounded by auxiliary circuitry and placed in a 'pod' which is connected to the host development system by a multiway cable and to the system under development by an integrated circuit header which has the same pin-out as the microprocessor. In use, the microprocessor integrated circuit is removed from the board to be debugged or examined and is replaced by the header. Now the development system can, via the auxiliary circuitry, reset, halt or interrupt the microprocessor and force it to either use memory in the development system, or that in the system under test. In a similar fashion, the memory and

input/output devices of the system under test can be accessed by the development system simulating the action of a processor executing a program that requires such an access. This process is shown diagrammatically in Figure 3.1. Now, using the software support facilities of the development system, the user can develop and assemble programs into a form that can be loaded into memory. In the first instance, these programs will usually reside in random access memory provided by the development system, even though they might eventually be placed in read only memory in the system under test. As there is a microprocessor of the relevant type in the in-circuit emulator pod, the programs can be executed before any hardware has been built, providing that they do not attempt to perform too much input/output. As the development system will usually provide sophisticated breakpoint facilities, all routines that call for input/output can be breakpointed so that the user may deposit values into the appropriate registers from the development system keyboard. In this way, many sections of the program can be debugged in advance of hardware availability. It has already been stated that microprocessor based hardware can best be debugged by the use of small test programs that 'exercise' the hardware so that faults can be traced easily with conventional tools. As soon as a hardware system has been attached to the emulation pod, the development system user can specify that an access to a particular location in the memory should be serviced by either the development system memory, or that of the system under test. This assigning of responsibility cannot be done for individual locations, but rather for groups of, typically, 256 locations at a time. The user can, therefore, leave a program in random access memory in the development system, rather than transferring it to read only memory in the system under test (with the attendant difficulty of changing the program each time an error is located), yet use memory and input/output devices in the system under test where possible and desirable. If the memory and input/output

devices will not work, a program can run in the development system that tries to access locations in the system under test, allowing the user to test logic states with any desired tool or the limited logic analyzer capability that is often provided by the development systems in conjuntion with the program trace capability. If this is done, the user will attach test leads to a selected number of points and run the test program. Then, for each bus cycle generated by the test program, a display will be generated that shows the value of the microprocessor data, control and address buses as well as that of the selected test points. A comparison can then be made between expected and obtained results, and the problem area can be restricted until rectification becomes possible. It is usual for any development system of this kind to provide EPROM programming capability, so that once programs have been tested they may be transferred to permanent residence in the system under test. If, at some future date, a fault in either hardware or software becomes apparent, the microprocessor can be removed and replaced by the emulator and critical locations examined or changed, and program execution traced. This is extremely convenient as it removes the necessity of providing terminal support and program monitoring software within the unit. This is desirable for, as mentioned previously, in embedded computer systems, system software is usually far larger than the task software.

The development systems described fall into two classes: the microprocessor manufacturer supplied, and those produced by independent companies. Those associated with a manufacturer, for example the Intel Intellec, Motorola Exorciser, National Starplex, could originally only support one processor. As the product families grew, the need to support more than one family on one machine became apparent. Now the above units support most of the named manufacturers products, if not with hardware emulation, at least with software development. The units produced by the independents are more

expensive, typically eight to ten thousand pounds compared with a manufacturer's version at five to eight thousand, but have the advantage of offering support for more than one manufacturers' products. These are the 'universal' microprocessor development systems or UMDS's. In particular, there are three major suppliers of universal development systems, Tektronix, Hewlett-Packard and Millennium Microsystems. Again, the range of facilities varies according to price, some systems offering hexadecimal keyboards and memory examination only, whilst others that are disk based offer complete and capable software development facilities.

Whilst the in-circuit emulators offer the most powerful approach to the problem of microcomputer education, there are three problems that must be overcome. The first major barrier to greater use is that of price. Systems that typically cost from five to fifteen thousand pounds per station, and will only offer emulation of one processor for that price, are acceptable in commercial development project terms, but not at any of the other development stages. The price of these systems is partly due to their great complexity, and could only be reduced if the complexity were to fall. The second problem is that the systems do not provide a minimal microprocessor board for experimental/educational use. The final problem associated with the development systems is that to date, all have been sold as a complete package. This has tied the user to the only source of software, the development system manufacturer. The software supplied is obviously written with the competent, full-time user of the development system in mind since there is very little support at any level. As the operating systems are all proprietary, the suppliers have not been inclined to supply details of the user software interface to the purchasers. This has inevitably restricted the ability of the user to write more software that suits their particular needs. With the above limitations, the use of in-circuit emulators in education laboratories has

been limited, one exception being at Bell Laboratories[10].

One very important feature associated with the in-circuit emulation based systems is the use of one microprocessor, within the development system, to assist in the examination of a target system. The power of this approach is further enhanced by the removal of all user supportive tools and complexity from the system under inspection. Whilst this is at its most extreme in the in-circuit emulators, where the microprocessor under inspection is also removed, the use of additional user supportive processors also appears in two other methods, the timesharing mainframe connected to a target processor by a serial link, and the dedicated mini/microcomputer connected by a serial or parallel link.

The use of two processors enables the combining of the best facilities presented by two different approaches. For example, the most common support arrangement is the provision of single board computers with a visual display unit and a serial link to a minicomputer[11]. The student uses the visual display unit to access the minicomputer and run the editors and cross assemblers or simulators that will create his program. Once the program is prepared, the minicomputer enters it, via the three way switch, into the microcomputer. The terminal then connects to the microcomputer and the student uses a monitor resident in the microcomputer to examine and execute the program which is now freed from the constraints of minicomputer simulators and can use any input/output facilities required. There are several advantages to be gained by this approach. With the removal of the editors and assemblers from the microcomputer to the minicomputer, the microcomputer system can become significantly simpler. The user also benefits from the better facilities that are available with minicomputers, such as simultaneous printing of files during execution of other programs. However, there are problems associated with this approach. The monitor program that resides

within the microprocessor must be in read only memory, and although it is reduced in size, it will still be a complex piece of software that must be written in the assembler language of that microprocessor. The monitor must be present to allow the user to examine and alter memory locations and control the execution of programs. The program can only be discarded if some other agency then becomes responsible for these functions. This implies the removal of software complexity by increasing the hardware complexity – an example of a hardware/software trade off.

The in-circuit emulator achieves this by the replacement of the microprocessor integrated circuit by a debug unit. Another method is to provide an externally controlled direct memory access channel into the system. Direct memory access is the input/output technique whereby the processor initializes a counter system that will provide an address at which incoming data is to be stored, or outgoing data to be found. Thereafter, whenever the input/output device requests a transfer, the counter system will take control of the processor bus, provide address and control signals appropriate to the participating memory location and manipulate the signals necessary to achieve a transfer to the input/output device. Once a transfer has been completed, the counter will increment and be ready to repeat the process without further processor intervention. This reduces the workload on the processor and enables the processor to undertake other work whilst also allowing a far higher peak data transfer rate to be achieved. Obviously, if direct memory access initialization is passed to the input/output device, (in this case, a minicomputer), the processor plays no part at all in such transfers and hence no program is required. An implementation of such a system between a Digital Equipment Corporation PDP11/20 minicomputer and a Motorola 6800 based microcomputer is described by Holdstock[12]. and to a Ferranti F100L based microcomputer by Selwyn[13]. The use of a minicomputer has several advantages.

As most minicomputer operating systems can support more than one user simultaneously, one machine can support an entire laboratory. Users and those responsible for software maintenance benefit from the well supported packages (such as editors) that are supplied with such systems. Inevitably there are disadvantages, particularly the fact that the minicomputer represents a large capital cost that is largely independent of changes in the number of working positions supported. Should the minicomputer fail, the entire system becomes inoperative until a repair is effected and it will be impossible to move the system for prototype or demonstration purposes.

Another major disadvantage is the method that must be used to implement the controlled direct memory access channel into the microcomputer. Typically, high performance multi-user minicomputers employ high speed, tight specification processor buses which are maintained by the manufacturers or their agents. This usually forces the interface to the target microcomputer to be implemented as an input/output device, using manufacturer supplied circuit cards to provide TTL compatible lines. For example, the two schemes mentioned employ the DR11C 16 bit input/ 16 bit output parallel port as an interface.

There is, however, an important difference between the use of a microcomputer as a peripheral and ordinary minicomputer input/output devices. Usually minicomputer input/output is byte access sequential, whilst block access may be either sequential or random. That is, the smallest addressable unit of data tends to be a block or group of bytes, such as a sector of data on a disk or a block of data on a magnetic tape. The data within these blocks is accessed in a fixed order that cannot be changed. The memory of a target microcomputer is unusual in that the data bytes, the contents of the target memory, can be accessed in any order and, indeed, will be required in a 'random' order. The same is true of the minicomputer memory and the addressing modes provided represent methods of accessing data held in such memories by

allowing the processor to perform calculations and output the results onto the address bus. If an input/output device is used as the basis of an interface unit, the address bus will only be capable of accessing that device since no peripheral requires the capability for byte addressing. This means that control of the addressed byte in the target microcomputer must be achieved by transferring data through the input/output device and this is the technique used by Holdstock and Selwyn. The sixteen output bits of the PDP11 peripheral port are split into eight data and eight control bits. For each word, the upper byte identifies the lower as being high/low byte of address or data. The resultant scheme is shown in Figure 3.2 and Table 1.

The work required of the minicomputer is excessive and fails to make use of the addressing modes provided by the minicomputer processor since the addresses are manipulated with the data oriented instruction set, resulting in a high software overhead associated with accesses to the target processor, thus increasing the complexity of software and limiting the top speed of data transfers.

The new method examined in this work is a synthesis of the in-circuit emulator and supportive minicomputer approaches. The suggested technique is to provide an interface between the microprocessor selected as the supportive unit and that used as a slave or target system such that the address, data and control lines of each can be translated to the timings and levels expected by devices attached to the other. This means that it will be possible for either processor to place upon its address bus a value that is recognised as requiring action by the memory or input/output devices associated with the other processor and the interface will automatically initiate a direct memory access cycle on the other processor bus. In other words, the two processor systems form an asynchronous, shared memory multiprocessor system with the processors involved coming from different manufacturers and families.

As the normal addressing modes of the processors will be used, there will be no software overhead associated with an access of the memory attached to the other processor and this reduces the complexity of the supportive software. The system under investigation appears as more memory to the supportive processor, so that program load and memory examination functions will be common to tasks related to both the supportive, and the supported, systems. Those responsible for the support of such systems will only have to write software for the supportive processor, offering the possibility of subroutine libraries providing commonly required functions. Students should derive the benefit of a comprehensive supportive software capability and simple targets for examination; those requiring prototype boards will be able to use simple cards designed for full expansion and with the ability to interface to a more supportive development tool. As the described unit uses a dedicated microcomputer per station, the cost is linearly related to positions provided, whilst the inclusion of the visual display unit function within the supportive processor further reduces costs.

As with both the in-circuit emulator and direct memory access channel based devices, this new technique relies upon the ability to transform bus control and timing signals from those generated by one processor to those required by another. The ability to transform such signals, and the general applicability of the technique is studied in the following chapter.

4. Bus Structures

This chapter examines the various bus control schemes found in microprocessor systems and assesses the generality of the bus to bus interface proposed in the previous chapter. The discussion will include most of the microprocessors families currently available and, as the design of microcomputer bus structures usually follows that of larger computers, a top range minicomputer, the Digital Equipment Corporation VAX, will be examined as a possible target microprocessor.

In attempting to classify the many methods used by manufacturers to attach support circuits to a microprocessor, it is often found that the fundamental necessity of the functions provided is ignored and discussions proceed on the basis of extant products. This does not provide a sound starting point for the analysis of possible future schemes, and, in this dissertation, the basic requirements of computer bus structures will be discussed initially and the examination of typical microprocessors will be in terms of the elements so covered.

Inevitably, microprocessor bus structures have been based upon those already in use for mainframe computers. Indeed one critic[14] described the microprocessor as 'Twenty years of architectural bungling concentrated onto one chip'. Usually the computer is split into four major elements, memory, input/output, arithmetic/logic unit and control unit. The last two share many components and are usually combined into the central processor unit (CPU) or microprocessor unit (MPU). In this scheme, the processor will request the next instruction from a list held in memory and will execute it, accessing additional memory locations to perform data transfers as necessary. The above scheme holds true for the common architectures, such as accumulator, register, stack and memory to memory, all of which have been used as the basis of microprocessor architectures.

The sequence of events is typically as follows. Firstly, the processor initiates a bus cycle by selecting a memory location to be accessed. This is most frequently done by outputting the binary address pattern onto a reserved address highway, although as microprocessors become more complex, the multiplexed address/data highway is becoming more popular due to package limitations. At approximately the same time, an indication is given to the memory devices as to the direction of the data transfer, i.e. processor to memory (write) or memory to processor (read). At some stage, an indication must be given that the address bus has become stable, since, if this is not the case, the transition period of the individual address bus lines could lead to false memory access. If the operation is a data/instruction read, the processor will then accept data from the memory. Two common methods are used: wait for a fixed period then take whatever is on the data bus, or wait indefinitely until the memory signals that data is ready. If the operation is a data write, the processor must present data to the memory on the data bus and either of the two methods described above may be used to control the transfer.

The essential elements of a memory or input/output transfer are, therefore, a means of selecting one of a range of possible locations, a means of establishing the direction of the transfer, indicators that address and/or data lines are stable and an indication of acceptance by the participating device. Although many methods have been used to provide the basis of a processor/memory protocol, all must provide the above features by one means or another.

Other controls provided by the microprocessors are not essential and are therefore open to greater variation. Common controls, though by no means universal, are: -

(a) An absolute method of gaining control of the processor. For example at

power on, typically called reset or restart, this control will force program execution to start at some pre-determined location.

(b) Means to interrupt the processor, that is to suspend execution of the current task in favour of one with a higher priority, usually associated with input/output. Interrupts can be provided as a single, or several levels of priority. In conjunction with the interrupt facility, some processors issue an interrupt acknowledge signal, so that the interrupting peripheral can identify itself by some pre-determined mechanism.

(c) A request that the processor ceases execution and 'halts'. This is often used as a means of allowing external access to devices usually controlled by the processor for direct memory access, or to synchronize the software with some external event.

(d) The bus request and grant signals are an alternative method of gaining control of the bus, as opposed to halting the processor. A bus request is an explicit indication of direct memory access activity which the processor will honour as soon as convenient, indicated by assertion of the bus grant signal.

(e) More recent processors have the capability for passing work onto a 'co-processor', responsible for high speed arithmetic or input/output. Data is frequently transferred to these devices by the main processor controlling the address bus, whilst the co-processor controls the data bus, and synchronization is therefore required.

(f) it is highly desirable, when debugging computer systems, to have an indication of the current state of the processor. A recent trend is towards the provision of status lines that indicate instruction or data fetch cycles, whether a normal or interrupt program is being executed.

As stated, not all the processors offer all the above facilities. However, in a system designed to provide a demonstrative and supportive tool for users, whatever features are available on the microprocessor should be

available for investigation. Therefore, any interface must be capable of both monitoring and activating all of the above lines where they exist, and, more importantly, should be capable of monitoring any transient effects associated with such activation.

Microprocessor buses can be grouped according to several different criteria. Possibly the most significant difference for the purpose of the present study is whether the bus is available, or is the entire microcomputer and support circuits provided on one integrated circuit? If this is the case, the only possibility is to provide full in-circuit emulation by using manufacturer supplied 'expanded' single chip microcomputers, where, to aid development, a special version with extra pins is made available. The next most important factor is voltage compatibility. Most microprocessor interface/buffer chips are TTL, and, as such, have strict voltage limitations. If an interface is to be provided to a PMOS or CMOS microprocessor system, the interface will have to be constructed from integrated circuits that were not designed as microprocessor supportive and are consequently less convenient and more expensive.

Assuming that the bus is both available and operates at the same voltages as that to which it is to be interfaced, there remain three areas which will require special attention. The first of these concerns the method of providing data and address highways. As provision of more pins on an integrated circuit increases the cost- there is a tendency to reduce the pin count where ever possible. However, if this is done, then the facilities that the microprocessor can offer will also be limited, that is, status lines will be removed or the number of interrupts reduced. To overcome the pin count limitation, pins can be made dual purpose, and, whilst any group of pins can be combined in this way, the obvious candidates are the address and data lines. If the address and data buses share the same pins then, for the first

part of any bus cycle, the pins carry an address, while, for the later part, the data associated with that address. The data and address lines are nearly always chosen for multiplexing, thereby giving the greatest possible return for the minimum extra complexity. One feature available when address and data buses are used in this way is reduced pin-out memory and input/output integrated circuits. Should control signals be multiplexed, they will not provide so great a saving due to their low numbers, whilst their diverse timings will further complicate the system. One notable exception is the Intel 8080, though the successor, the Intel 8085, has moved to a multiplexed address and data bus. As part of any interface, it will be necessary to demultiplex and multiplex the buses as required.

Related to the interfacing of a multiplexed bus is the question of the relative widths of the address and data buses of the two microprocessors concerned. In this study, the Z80 is used as the supportive processor. This has an eight bit data bus with a sixteen bit address bus. When the Z80 is interfaced to another microprocessor, it will have to be able to access any portion of the memory of that microprocessor. Should the memory be sixteen bits wide, the Z80 bus must be capable of accessing both the upper and lower bytes of the data bus, whilst if the other microprocessor has a twenty bit address bus (1 megabyte memory space), the Z80 should be able to access any location within that space.

Finally, the control signals must be derived. As previously stated, all systems need an address validation signal, a data transfer direction indicator and a data validation signal. There are many different methods of generating these signals which are frequently combined in some way. As these signals will often be used to drive internal latches on various integrated circuits and, as they will be conditioned by passage through high speed bipolar logic, rather than the slower MOS logic of most common memory chips, it is critical

that these signals are to specification.

There are several frequently occuring arrangements of control signals that can be discussed without reference to specific families. The first of which is detailed in Figure 4.1, where the bus is shown for both read and write cycles. When the microprocessor is writing, the address validation signal validates data and control signals as well. When reading, only address and control lines are validated and the responsible peripheral *must* provide data within a pre-fixed period of the fall of the address validation signal if reliable data reads are to be ensured. The other method of ensuring validation is the employment of separate read and write lines used to validate data. Such a scheme is shown in Figure 4.2.

The present study has been restricted to eleven microprocessor families.

4.1 The Zilog Z80 Microprocessor

The Z80 microprocessor has sixteen address and eight data lines. Compatible at the machine code level with the Intel 8080 and lacking just two of the Intel 8085 instructions, it is currently the most widely used of the eight bit processors. The Z80 has a single phase clock and the manufacturers provide parts with maximum clock speeds of 2.5, 4 and 6 MHz, with all bus timings in similar ratios. As the Z80 has separate Input/Output and memory address spaces, two lines are used for address validation and selection. $\overline{\text{MREQ}}$ selects the memory address space and is asserted only when the address bus is stable, whilst $\overline{\text{IORQ}}$ performs the same function for the input/output address space. The two signals $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are used to control data transfer direction, and, whereas $\overline{\text{RD}}$ follows $\overline{\text{MREQ}}$ and $\overline{\text{IORQ}}$ with the participating support device responsible for supplying data in a fixed time or requesting additional time (see $\overline{\text{WAIT}}$), the $\overline{\text{WR}}$ signal acts as a data validator thereby allowing time for both the address and data buses to settle before becoming active (note that $\overline{\text{MREQ}}$ and $\overline{\text{IORQ}}$ have already validated the address). Should any device not be

able to provide or accept data within the required time, it may request additional clock cycles by asserting the $\overline{WAIT}$ line. There is theoretically no limit to the number of wait cycles that may be inserted into a bus cycle, but one of the most powerful features of the Z80, the ability to refresh dynamic memories, will be defeated if the $\overline{WAIT}$ line is active for too long. As one location is refreshed per op-code fetch, and there are 128 locations that must be refreshed at least every two milliseconds, this represents a maximum wait cycle of

2000 - (128*longest instruction time*time/cycle)

where the longest instruction time is 23 cycles. This provides figures of 859.2 microseconds, 1264 microseconds and 1509.3 microseconds for 2.5, 4 and 6 megahertz processors respectively.

In practice, these figures will only be approached by large scale direct memory access transfers since, with typical accesses being 450 nanoseconds, the shortest time represents 1700 memory access periods. It is also possible to force the processor off the bus by asserting the bus request signal $\overline{BUSRQ}$. When the Z80 has finished executing the current bus cycle, $\overline{BUSAK}$ will be asserted to indicate the availability of the buses. Maximum bus holding times apply as before as the processor is prevented from accessing the dynamic random access memory for refresh purposes.

All devices to be found in Z80 systems operate on the four signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$ and these four are the only signals that must be within the precise constraints of the Z80 timings when the bus is to be driven from an external source. For most devices, the constraints imposed upon these signals will be no more than the observance of minimum timings for transfers between the asserted and non-asserted conditions. This suggests that an interface to a Z80 should be easy to implement. In fact the task is somewhat easier, considering that the supportive processor is also a Z80!

4.2 The Motorola 6800 Microprocessor

The bus timings associated with the Motorola 6800 family differ markedly in approach from those used with the Z80. Whilst the Z80 uses one clock signal that is not directly responsible for address and data validation, the Motorola 6800 uses two clock phases, one of which is the reference for all bus activity. The first phase ($\phi 1$) is used by the processor only, and is generally not required elsewhere in the system. The second phase ($\phi 2$) is used as the primary address/data validator. The use of the primary system clock as an address/data validator has two important consequences. If the processor is engaged on internal activity, for example, address calculations, a clock cycle will occur in which the address on the bus has no relevance. If an input/output device is referenced by this false address, it is possible that a status flag will be affected incorrectly as many flags in input/output devices are automatically reset by a processor read. To correct this problem, it is necessary to provide a cycle invalidation signal. This is done in the form of the 'Valid Memory Address' or VMA signal. Under normal circumstances, VMA is always active (high) and, only when the next cycle is a processor internal cycle, is VMA taken low. Normally VMA is not defined to be three state, that is, it is always driven by the processor. During direct memory access cycles it is driven low by the processor, but as VMA should be used in all address decode circuits, it is necessary to provide an external control over VMA to ensure that external devices can access memory and input/output correctly. One possible method is that adopted by Motorola for later 6800 compatible processors. Here VMA does not exist and, for internal cycles, the Motorola 6809 reads address $FFFF_{16}$ (the reset vector). VMA could be used in the same fashion, thus obviating the need to transmit it round the system. Such a scheme is shown in Figure 4.3

The second consequence arising from the use of $\phi 2$ as an address validator is that the only method of interfacing to slow memory devices is to stretch the primary system clock. This requires considerable circuitry to achieve correct restarting of the oscillator under all conditions, and this circuitry is usually provided by a clock module from Motorola. These units require that a request for a $\phi 2$ stretch should be issued during the preceeding $\phi 1$ high period. This can only be done once the address that is to be accessed is known, which unfortunately is validated by $\phi 2$, and the phases are none overlapping. It therefore becomes necessary to generate further address validation signals that operate independently of $\phi 2$.

It should also be apparent that any clock devices that derive a reference from the system clock will also be affected by the stretch of the $\phi 2$ signal.

The Motorola 6800 is a dynamic device, that is, internal memories require periodic clock cycles to maintain data. This implies that there is a lower limit to the clock frequency that can be used or caused by clock stretching. This limit is 100 kHz or 10 microseconds per period and if the clock is held for any longer, correct operation is not guaranteed. The limitation on clock period is of major importance in a multi-processor configuration. If the 6800 requests a memory access cycle that involves the second processor, that cycle must complete within 10 microseconds.

Any processor that accesses a Motorola 6800 system will be constrained by the action of the 6800 clock. This is an interesting feature of the 6800 direct memory access scheme, as although data, address and direction are specified by the DMA device, the precise starting point of each transfer is still fixed by a processor related unit, i.e. the clock. As the clock is not a three state signal, it will not be possible to provide an external clock generated by the second processor.

These considerations make the Motorola 6800 an extremely difficult

processor to interface in the manner suggested and the interface will be discussed in greater detail in Chapter 8 to demonstrate the resolution of these problems.

4.3 The Intel 8085 Microprocessor

As the Z80 was the Zilog up-grade of the Intel 8080, so the 8085 was Intel's up-graded processor. In almost all applications, the 8085 is the preferred eight bit processor provided by Intel.

Whilst the two processors share a common instruction set, the 8085 has two new instructions not provided by either the 8080 or the Z80. These are the RIM and SIM instructions which handle the single bit input/output ports provided by the 8085 processor.

Although the software facilities of the 8080, Z80 and 8085 are very similar, the hardware schemes are not. However, the 8085 bus interface signals are far closer to those of the Z80 than are those of the Motorola 6800. The main feature to be noted is that the 8085 uses a multiplexed address and data bus, the lower eight lines of the sixteen bit address bus act as the data bus. As described earlier, for the first part of any bus cycle, the bus contains an address. Once the address is stable, the address latch enable (ALE) signal is asserted to latch the address into peripheral/memory devices. The 8085 has two address spaces, one for peripherals (I/O space) and one for memory. These are selected by the action of the IO/$\overline{M}$ signal. Address and data validation are performed by the $\overline{RD}$ and $\overline{WR}$ signals. As in the Z80, slow memory/peripheral devices are interfaced by using the 'ready' signal operating independently of the system clock. The 8085 provides two status lines that provide the current processor status, differentiating between halt, read, write and instruction fetch states.

The similarity of approach to the 8080, also adopted by the Z80, renders the interface much simpler than that required by the 6800. The major

noteworthy point is that most 8085 support chips expect and require the multiplexed bus and provision must be made to allow the use of these devices.

4.4 The Mos Technology 6502 Microprocessor

The Ziiog Z80 and Intel 8085 represent up-dated versions of the Intel 8080, and the 6500 family, of which the 6502 is the most powerful member, was produced in response to the introduction by Motorola of the 6800. Unlike the Z80, which executes all the 8080 instructions but is not hardware compatible, the 6502 executes a different instruction set from that of the 6800, but is similar in the bus structure that is employed.

The processor has an address bus of sixteen lines with an eight line data bus, and, as in the Motorola M6800, data direction is determined by a composite R/$\overline{W}$ line. The major differences between the bus structure of the two devices relates to the method used to interface to slow memories. As already discussed, the Motorola M6800 has no circuitry to accomodate slow memories, and as a consequence, the processor can only be used with slow memories by stretching the $\phi 2$ clock high period, effectively slowing the processor temporarily. The 6502 has a ready line which, when taken low, causes the processor to insert additional clock periods into the bus cycle. As the $\phi 2$ clock is no longer always high during the active part of bus cycles, it cannot be used as an address validator. In fact, the 6500 family as a whole lack address validation, and the philosophy employed is that, if an address is present on the bus long enough to be recognized by memory devices, the appropriate devices should respond.

The cycle invalidator of the 6800 (VMA) is also absent, any internal cycles ensure that the R/$\overline{W}$ line is held high, and a read cycle takes place. A further difference is to be found in the approach to direct memory access. Whilst the 6800 can be halted and access gained to the bus, the 6502 must be held by the application of the READY signal. This implies that the processor

is halted during a bus cycle, when the address, data and control buses will all be active. This is, in fact, the case and, in consequence, there is no control signal available that can be used to force the processor into a high impedance state. To achieve direct memory access, it is therefore necessary to surround the processor with three state buffers, and whilst this increases the chip count, the resultant external control gained over the buffers can simplify the task of direct memory access circuitry. The last major point concerning the 6502 is that the READY line will not insert wait states into a write cycle and if memories are used that have a long write cycle, the address and data lines must be latched. One additional status signal is provided on the 6502, like the Z80 and 8085 and indication is given whenever the processor fetches an instruction. This output is provided by the SYNC signal.

4.5 The National Semiconductor INS8060 (SC/MP II) Microprocessor

The designers of the SC/MP (Simple Cost effective MicroProcessor) produced the first microprocessor aimed at the slave processor market. The processor is designed so that several SC/MPs may be linked together to form a multi-processor system, or a single SC/MP may be attached to the bus of another processor to share the workload. As the technique under discussion is based upon examination of a processor by making it a slave to a supervisory microprocessor, the control signals of the SC/MP are of the right form for use in this context.

The most notable feature of the SC/MP bus interface is that the SC/MP does not expect to be bus master. With all other processors discussed to date, the processor owns the bus and *grants* access to the peripheral performing direct memory access. With the SC/MP, any device can *force* the SC/MP off the bus, even in the middle of an instruction. To provide this facility, the SC/MP is provided with three bus arbitration signals which are unique amongst the

eight bit microprocessors. These are NBREQ, which indicates to the bus controller that the SC/MP requires a bus cycle, NENIN which is used to indicate to the SC/MP that the bus is available, and NENOUT which is used by the SC/MP to indicate to lower priority bus users that the bus is available, but not being used by the SC/MP. These three signals allow easy implementation of a supervisory/slave microprocessor system.

The other bus signals are conventional and consist of an eight bit data bus with a sixteen bit address bus where the upper four bits are multiplexed onto the data bus at the start of each bus cycle. There is a written data validator (NWDS), a data read signal (NRDS) and an address validator and multiplexer control (NADS).

Although the SC/MP can address sixty four kilobytes of memory, most applications using a SC/MP restrict the address space to the four kilobytes that are available without address demultiplexing.

Due to the bus arbitration scheme mentioned above, there is no need for a halt state to enable direct memory access by other bus devices. However, where several lower priority devices are present that may occasionally require a burst of data transfers without interleaved SC/MP cycles, the processor may be held inactive by use of the CONT signal. When low, the processor ceases activity at the end of the current instruction and the buses are floated. This method does not halt the SC/MP indefinitely. Should an interrupt occur, the processor will execute the first instruction of the interrupt routine, which may be used to reacquire bus mastership.

4.6 The Texas 9900 Microprocessor

The Texas Instruments 9900 was the first sixteen bit microcomputer to appear on the market and was designed by Texas to be a single chip implementation of the Texas 990 minicomputer processor. As such, the bus

structure and processor architecture that the 9900 embodies was designed to be compatible rather than innovative. The processor is packaged in a sixty four pin integrated circuit, allowing the use of a non-multiplexed sixteen bit data bus with a fifteen bit address bus. This means that the 9900 can address thirty two kilowords of memory. Unlike most later sixteen bit processors, a word cannot be addressed as two separate bytes and all byte operations consist of word read, byte up-date, word write cycles. Any transfers managed by an eight bit supervisory processor must therefore either operate on the same read/modify/write basis, or gather two eight bit values before undertaking one sixteen bit write. It follows that either approach will increase the complexity of that part of the supervisory processor/slave processor interface concerned with data buses. Although the processor requires a four phase clock, this is not used for data or address validation as address validation is supplied by the $\overline{\text{MEMEN}}$ (MEMory ENable) signal, whilst $\overline{\text{WE}}$ validates data to be written to memory. DBIN (Data Bus IN) is used by the processor to indicate that data should be placed on the bus by a memory device, so that it functions as a read signal. Accomodation of slow memories is performed by the READY signal, which is used by slow devices to request insertion of additional wait states into the bus cycle. Direct memory access is accomplished by the use of $\overline{\text{HOLD}}$ and $\overline{\text{HOLDA}}$ (HOLD Acknowledge) signals, all relevant processor control signals being floated before $\overline{\text{HOLDA}}$ is asserted. The IAQ (Instruction AQuisition) signal is used by the processor to indicate that an instruction fetch cycle is in progress.

Apart from the problem generated by the word only access of the data bus, the input/output scheme used by the 9900 microprocessor (also a consequence of its derivation from the 990 minicomputer) is so unlike that of any other processor as to require specific and unique circuitry to enable another type of processor to access it satisfactorily. The input/output address space is

organized as four thousand and ninety six single bit locations, data being sent to these locations by the processor's single bit output bus, CRUOUT and validated by CRUCLK. Data is returned to the processor on CRUIN. Programs within the 9900 access these lines by giving an address for the transfer and the number of bits (1-16) to be transferred, the processor acting as a parallel to serial converter for these operations. Any processor that controls a Texas 9900 system must either access the 4096 bit locations as such and assemble the data into groups using software, or use special hardware to accomplish the same object. The exact operation of such hardware is quite important if the 9900 has many input/output devices on the CRU system, as the ability of the 9900 to limit the size of the transfer to individual bits enables the close packing of the input/output devices in the address space. Should the design of the processor/processor interface hardware not allow control of transfer size, other input/output devices might be accessed incorrectly.

Although this chapter deals primarily with the hardware aspect of processor to processor interfaces, the Texas 9900 has one software architectural feature that simplifies the software of the supervisory processor. Unlike other microprocessors discussed, the 9900 uses memory based registers, that is, the only registers within the processor itself are the program counter, status register and a workspace pointer that identifies the section of memory currently being used as general purpose registers R0 to R15. There is no stack pointer as such, the workspace pointer is saved in temporary storage, reloaded with a new value, and the original value is saved in the new R13. There is now a new set of registers for use by the interrupt routine or subroutine. This is a useful feature in that the supervisory processor will require access to the registers of the slave microprocessor and in most cases this will require a small program run by the slave processor to dump registers

to a fixed location in memory for examination by the supervisor. With the Texas 9900, an interrupt will leave all the current registers open for examination in memory as part of normal operation.

4.7 The Intel 8086 Microprocessor

The Intel 8086 was the first of the second generation sixteen bit processors to be introduced. Unlike Texas and the Texas 9900, the Intel 8086 was not the first microprocessor product from Intel, the 8080 and 8085 were both in volume production and use. Due to this, the Intel designers had the opportunity to modify any of the features of their eight bit devices based upon market experience. In fact, although the 8086 is not object code compatible, it is possible to use programs that take 8080 assembly language source, and convert it to 8086 assembly language source. The hardware scheme is broadly similar to that of the 8085. The 8086 has twenty address lines (a one megabyte address space) which are multiplexed with sixteen data lines and four status lines.

An important and novel feature of the 8086 is the provision for two modes of working. A line into the processor can be used to select a pinout appropriate for a minimal system that uses few memory interface components, or a maximal system, where the processor provides more status information in encoded form, relying on external support chips for the decoding of this information and the provision of some basic bus control signals that are provided by the processor itself when in minimal mode. In this study, the minimal case is more appropriate as it offers reduced circuit complexity and therefore the description of the 8086 pinout and timings will refer only to this configuration.

There are two address spaces provided by the processor, memory at one megabyte and the input/output at sixty four kilobytes. One of these two spaces

is selected by the M/$\overline{IO}$ signal, high selecting memory, and low input/output. The address is validated by the ALE (address latch enable) signal which is used to store the multiplexed bus into latches to provide a demultiplexed address bus. The Intel 8086 bus is organized as two eight bit banks, with A0 selecting the high or low byte of a sixteen bit word. If A0 is low, the low byte is accessed. If, however A0 is high, only the high byte is accessed. For sixteen bit read and write access, A0 is held low, thus selecting the low byte, and another signal $\overline{BHE}$ (bus high enable) is also asserted, forcing selection of the high byte. Thus the additional bus access flexibility over the Texas 9900 has been gained at the expense of another address line and this will be repeated for all sixteen bit data bus, byte accessing microprocessors. Data direction control is provided by the $\overline{RD}$ and $\overline{WR}$ signals with $\overline{WR}$ acting as the data validator. A further facility which is given two pins in minimal mode, provides for the direct control of a data bus buffer. The signals DT/$\overline{R}$ and $\overline{DEN}$ are used to control the direction and output enable of an 8286 bus transceiver. Whilst these signals can be derived easily from $\overline{RD}$/$\overline{WR}$ and bus grant signals on an non-multiplexed processor bus, the dual purpose data highway requires slightly more complex timings, which is here handled by the processor directly. Two signals are used to control direct memory access, HOLD for requesting the bus and HLDA (hold acknowledge) for the processor to indicate bus available. One signal which is not present in the minimal mode, but which is relevant to the examination method suggested is the LOCK signal. Designed for use in multi-processor environments, LOCK indicates that a semaphore operation is occuring. That is, the processor requires to read, modify and write a memory location and no other device is to be allowed access to the bus between these events. This signal is used where, for example, two processors share a common resource such as a DMA controller, memory area or arithmetic processor. In such cases, the processor gain ownership of the

resource by testing a memory location, and if it indicates that the resource is currently unassigned, changing the contents of that location to indicate their use of the resource. The lock signal prevents each processor from simultaneously reading the location and identifying the resource as free, before both writing a value into the location, claiming the device. This is the first aid to the *programming* of multi-processor systems. Another innovative feature of the 8086 is that a software single step facility is provided on board the processor. This allows a program to execute another section of machine code an instruction at a time, with the processor automatically passing control back to the debugger. The provision of this facility in the 8086 and later processors is an acknowledgement by the manufacturers that software development costs are very high and that the sixteen bit machines are intended for use in an environment of continual software development and changes, that is, minicomputer replacement, rather than simple control tasks where the program, once written, is fixed.

4.8 The Zilog Z8000 Microprocessor

The next second generation sixteen bit processor to be released was that produced by Zilog. As the Intel 8086 has a configuration pin to enable selection of a minimal or maximal configuration, so Zilog offer a similar choice, but is more drastically organized, as either the forty pin Z8002 or the forty eight pin Z8001. The internal organization of the two processors is also different. The Z8002 can address thirty two kilowords, each of which can be accessed as two individual bytes, and is designed to run one task only. The Z8001 can access sixteen megabytes of memory which is also word organized and byte accessible, but the processor is designed to run several different and unconnected tasks presided over by a supervisory program. This is a typical multi-tasking minicomputer organization, where any instructions that can

either halt all programs or directly affect input/output devices is available only to the supervisor program. Supervisor mode is entered when any illegal instruction is encountered, when an interrupt occurs or when a program requests an action from the supervisor. Entry to supervisor mode from illegal instructions takes the form of a software interrupt, the supervisor program taking appropriate action.

When considering the minimal/maximal configuration, there is a clear distinction between the Intel 8086 and the Z8000 family. The Intel 8086 configuration merely affects the number of status signals available, while the Z8001/2 choice affects the addressing space available. As it is possible to duplicate all the features of the Z8002 with the Z8001, yet retain the greater address space, this study refers to the Z8001 exclusively.

The Z8001 has a multiplexed sixteen bit data and address bus. Validation of the address is performed by $\overline{AS}$ and written data is validated by $\overline{DS}$, whilst a single R/$\overline{W}$ line is used to select data direction. A further change from the Z80 is the use of $\overline{MREQ}$, the memory request line. On the Z80 $\overline{MREQ}$ acts as an address validator for the address space only. On the Z8000, $\overline{MREQ}$ low selects the memory space, whilst $\overline{MREQ}$ high selects the input/output space. Both these changes are designed to reduce the pin count of the processor. As with the $\overline{BHE}$ signal of the Intel 8086, B/$\overline{W}$ (byte/word select), in conjunction with A0, allows accesses of the memory to be low byte only, high byte only or both bytes (word access).

The sixteen address lines only provide access to the sixty four kilobytes available to the Z8002. Therefore, to increase the address space available to the Z8001, Zilog use the concept of segmentation, that is the the address map is seen to consist of 256 segments, each of sixty four kilobytes. Whilst this has a significant effect on programming technique, there is no reflection of this in hardware other than a name change. The segment number lines of the

Z8001 can therefore be considered to be high order address lines. Indeed, a similar programming environment occurs within the Intel 8086, but Intel draw no distinctions between segment and address lines. The processor has a standard wait input for accomodation of slow memories, whilst a $\overline{\text{BUSRQ}}$, $\overline{\text{BUSAK}}$ scheme is used to gain access to the bus for direct memory access purposes. Bus and wait requests must be of limited duration, for like the Z80, the Z8001 is capable of refreshing dynamic memory, this being achieved by regular bus cycles accessing an incremented address. Unlike the Z80, these cycles have no special properties other than a different value on the status lines. To identify to external devices the function of the current bus cycle, four status lines are available, whilst two signals are devoted to the multi-processing support role of identifying when the processor has entered supervisor mode (the N/$\overline{\text{S}}$ signal) and, when an illegal segment has been accessed, the $\overline{\text{SEGT}}$ line. This line is effectively a reserved interrupt for use with the Z8000 series memory management circuits, which are not used in this study.

The $\overline{\text{STOP}}$ signal of the Z8000 is used to provide single stepping of programs under development and examination. Given that the Intel 8086 has no provision for multi-tasking as the Z8001 has, the Intel software debug method is more suitable for a multi-tasking environment than the hardware method adopted by Zilog. The $\overline{\text{STOP}}$ signal will halt the processor at the end of or during the current instruction, depending on type, and force the processor to continually refresh memory until the $\overline{\text{STOP}}$ signal is taken inactive. To assist in multi-processor arbitration, the Z8000 has a single bit input, single bit output port built into the processor and controlled by special instructions. This is not as sophisticated as the SC/MP concept of multi-processor arbitration, consisting as it does of only a software communication device. It, therefore, has no relevance to this study, as a similar scheme can be

achieved with any processor by the use of a reserved memory location.

4.9 The Motorola M68000 Microprocessor

Although the M68000 uses a sixteen bit data bus, it was the first microprocessor to be released that used a thirty two bit internal organization. The address bus has twenty four lines which, due to the sixty four pin package, are not multiplexed. A major bus interface innovation of the M68000 is the use of an asynchronous bus. The address bus A1-A23 generates a word address, although software uses byte addressing. The lower/upper/both byte selection is performed by a transformed A0, the $\overline{UDS}$ (upper data strobe) and $\overline{LDS}$ signals. For read cycles, $\overline{UDS}$ and $\overline{LDS}$ are asserted at the same time as the address validator ($\overline{AS}$), $\overline{UDS}$ and $\overline{LDS}$ acting as written data validators. The asynchronous bus scheme requires that the processor should place the buses into a stable state for data transfer, and then wait until the participating memory devices indicate the completion of the data transfer with the data transfer acknowledge ($\overline{DTACK}$) signal. This is a reversal of the more usual technique where the processor assumes data will be ready within a fixed time unless held by a wait signal. The use of this technique allows the readier implementation of circuits to guarantee that bus cycles only access valid memory. That is, memory that exists will assert $\overline{DTACK}$, but if an access occurs to a location with no corresponding physical memory circuits, $\overline{DTACK}$ will not be asserted. There is thus no possibility of a read from a invalid memory location being used by a program as valid data. Should $\overline{DTACK}$ be the only signal used to terminate a bus cycle, an access to an invalid address would effectively freeze the processor, which would be waiting for $\overline{DTACK}$. To overcome this, the M68000 is provided with a bus error input ($\overline{BERR}$), which can be used by external circuitry to terminate an access to a non-responding location.

Whilst the above facilities could be implemented using a wait and bus

error scheme, the requirement for a positive response from participating memory devices, rather than the negative response of action only if more time is needed, provides protection for the system from boards that develop faults in service.

The M68000 contains a bus arbitration scheme similar to that of the SC/MP. Implemented in hardware is a protocol that can be used to transfer control to another device. All possible bus masters are connected to the Bus Request input of the processor. When low, this signal informs the processor that a device requires a bus access, and the processor replies with the Bus Grant signal that indicates that the bus will be available on completion of the current bus cycle. When $\overline{AS}$, $\overline{DTACK}$ and Bus Grant ACknowledge ($\overline{BGACK}$) are all inactive, another bus master may assert $\overline{BGACK}$ and remove bus request, thus claiming mastership of the bus. As soon as $\overline{BR}$ is inactive, the processor will remove bus grant and wait for $\overline{BGACK}$ to become inactive, whereupon the processor is once again master. It can be seen that this arbitration scheme is the most comprehensive to date.

To allow the use of the earlier eight bit M6800 peripherals with the more complex bus protocol of the M68000, an input is provided ($\overline{VPA}$) which is used to indicate that the address placed on the bus will be acknowledged by M6800 peripheral devices. The processor will then generate timing that is consistent with that of the M6800. In particular, the requirement that data transfers take place during the $\phi 2$ high period, is provided for by the E signal generated by the M68000.

Also provided by the M68000 are three status outputs to indicate whether the current bus cycle is an interrupt acknowledge or a user/supervisor data/instruction fetch. Like the M6800, the M68000 has a halt line, but here it is both an input and an output. If the processor receives a bus error signal, it attempts to read a vector giving the program address to which

control should be passed. If a bus error is signalled for the vector address, the processor halts and indicates this through the HALT line. Should an external device wish to halt the processor, a low level placed on the halt line will force the processor to halt indefinitely.

4.10 The Motorola M6809 Microprocessor

The M6809 is the latest of the eight bit processors to be released. Based largely on the M6800 architecture, the changes introduced have largely been directed at improving the facilities available in software. Although the same input/output devices designed for the M6800 can still be used with the M6809, the bus interface has been modified, clearly in response to the strengths and weaknesses of the M6800 bus.

The M6809 has sixteen address lines and eight data lines and uses a composite R/$\overline{W}$ line as does the M6800. Unlike the M6800, two versions of the M6809 are available, one for small systems that has in built crystal drivers and clock logic, another that uses external clock circuits and provides additional processor status information. Instead of the two non-overlapping clock phases required by the M6800, the two clock inputs are in quadrature, one phase being a direct replacement of $\phi 2$, (the primary bus validator of the M6800) known as E, and the other known as Q. Addresses are now guaranteed valid on the rising edge of Q, whilst written data is stable before the falling edge of the Q clock. No such signals are available on the M6800. The M6800 cycle invalidator (VMA) no longer exists, as the M6809 will always perform a valid bus read from location $FFFF_{16}$ (the reset vector) during internal processor cycles.

For the version with the internal clock logic, slow memory is accomodated using the MRDY line, which, although appearing to act as a wait state requester, actually stretches the internal version of E, and is thus subject to the ten microsecond limit, associated with cycles on the M6800. Direct

memory access transfers are controlled either with the $\overline{BREQ}$ or TSC lines, dependent on the clock option in use. The $\overline{BREQ}$ signal acts in the same way as MRDY, but disconnects all processor drivers from the bus so that address, data and control buses float, whilst the external versions of E and Q are maintained to allow the DMA cycle to be timed. Internally, this operation is observed by the processor as a long memory access, so a total bus cycle limit of ten microseconds is still enforced, part of which is required to complete the processors bus request. For the external clock version of the M6809, the three state control line (TSC) is used to force the processor off the buses and, in addition, two other status signals are made available. These are last instruction cycle (LIC), which gives advance notice that the next bus cycle will be an instruction fetch, and the processor BUSY signal, which can be used in multi-processor configurations to indicate a collection of bus cycles that must proceed uninterrupted by other processors or DMA devices accessing the bus.

Both versions of the processor provide two status signals which indicate the current status of the processor and the type of bus cycle being performed. The bus available (BA) and bus status (BS) signals thus differentiate between processor running, processor halted or bus granted (the processor action in each case being identical), interrupt acknowledge or SYNC acknowledge. The SYNC concept allows the processor to wait for an interrupt and either continue normally or perform the interrupt routine when an interrupt occurs. The halt line of the M6800 has been retained unchanged on the M6809.

4.11 The Ferranti F100L Microprocessor

The F100L is a sixteen bit microprocessor from Ferranti Ltd. The major market for F100 devices is the UK armed services, as the F100 is one of the few microprocessors that meets British Standard 9000, for which one requirement is that the devices be of UK manufacture. The high speed implied

by the bipolar nature of this family is offset by the organization of the arithmetic unit within the microprocessor, which is serial in form. The bus protocol of the F100 is significantly different from, and more complex than, that of the other microprocessors described and usually requires the assistance of the memory and peripheral interface integrated circuits that are also produced by Ferranti to support the F100. In certain respects, it is more accurate to describe the bus between the F100 and the support devices as a highway internal to the processor, and to regard the family as a multi-chip processor with a more orthodox bus structure, that which is provided by the support devices.

The F100 uses a multiplexed address/data bus of sixteen lines. As the processor accesses only thirty two kilowords of address space, the remaining address bit acts as part of the read/write logic, basically, an active low read line. To transfer data from the processor or direct memory access devices (referred to by Ferranti as the 'active' devices) to memory or input/output ('passive') devices, requires four control lines, $\overline{\text{J(Acv)}}$, $\overline{\text{J(Pas)}}$, $\overline{\text{K(Acv)}}$, K(Pas). The action of these lines is complex in description and the three possible types of cycle (read, write, read/modify/write) are shown in Figures 4.4 to 4.6, with the action of each line and edge marked.

Whilst the top bit of the address bus acts as a $\overline{\text{READ}}$ line, there is another line $\overline{\text{WText}}$ (write to external device) and of the four possible combinations of these two lines, three are used to indicate read, write or read/modify/write cycles, this action also being shown in Figures 4.4 to 4.6.

Direct memory access requests are more straightforward, with the requesting devices asserting $\overline{\text{DMARq}}$, bus availability being indicated by the processor asserting $\overline{\text{DMAAccept}}$. Both these lines pass through the support devices in a daisy chained fashion, so that a complete bus arbitration scheme is provided for the system, with priority of access being fixed by the

position of a device in a chain. Of interest in relation to DMA, and other bus cycles is the ability of the F100 and all the supportive devices to monitor bus cycles and independently decide if a cycle has failed. An external RC timing circuit can be attached to all family members which may be used to set a bit in the status word of that device if any cycle exceeds the time constant of the RC circuit. As well as being used by the device that detects the error to determine future actions, the status bit is available as an output line so that action can be initiated by failure recovery hardware if desired.

Interrupts to the F100 are of two types, non-vectored and vectored. Both use the Program Interrupt Request ($\overline{\text{PgItRq}}$) line, but in the case of vectored interrupts, the $\overline{\text{ExtLdPgCt}}$ (external load of program counter) line is used to force the acceptance by the processor of a new program counter placed on the bus. The two cases are shown in Figures 4.7 and 4.8. In each case, the $\overline{\text{PgItAccept}}$ line provides both a positive acknowledgement of the receipt of the interrupt, as well as temporarily locking out further interrupts until completion of the current transfer of control to the selected interrupt routine.

The F100 also has provision for the attachment of special function processors to handle the work load, for example, in the case of multiplication and division the F101 processor is used. Various instructions are reserved for external devices and, upon detection of such an instruction on the bus at the same time that the instruction fetch line ($\overline{\text{IRd}}$) is asserted by the processor, processing will cease. If a special function processor exists that recognizes that instruction, it will assert $\overline{\text{ExtFnAccept}}$ to indicate that the instruction has been accepted for processing, removal of $\overline{\text{ExtFnAccept}}$ allows the F100 to continue. Should no special processor be available, the bus will time out as described above.

As can be seen, the F100 bus structure is extremely cumbersome if

considered as a processor to memory bus, although it does allow for extremely reliable systems to be constructed. The provision by Ferranti of memory interface devices implies that the processor is not complete and that the bus is for use between various processor elements, one of which is a memory interface. As the interface components are so vital to the correct operation of the F100, they will be described in detail.

The interface set consists of one F111, which is responsible for the processing of control signals from both the processor and the interfaced device, while the processing of data and address information is the responsibility of the F112, of which two are required. Several interface sets may be attached to one F100, and the sets may be configured into several different arrangements to meet varying interface requirements. An ability possessed by the interface set in all configurations is the provision of additional bus drive capability so that more devices may be attached to the F100. The three major configurations of the interface set enable it to perform address decoding for memory devices, provide direct memory access capability, vectored interrupts and address decoding for input/output devices and detect the presence of an instruction on the processor bus that must be referred to a special function processor. The selection of one of the above configurations is performed by the connection of several configuration pins on each device to either zero or five volts. Therefore, each interface set can only perform one of the above functions once in circuit. A further device from the support family, the F113, can be used to further simplify the timing of the bus by the generation of address strobe and data buffer control signals, and so allow the direct connection of memory devices to the processor without the requirement of a full interface set.

In the peripheral interface mode, a direct memory access address counter is provided within the interface set, and this can be loaded either by the

peripheral chip, or by the F100. Similarly, provision is made for a value to be forced onto the processor bus during a vectored interrupt. If two or more F100 systems are to be connected, as is the case when multiply redundant processors systems are used to increase reliabilty, two interface sets configured in peripheral interface mode may be placed back to back, each processor gaining access to the memory and input/output devices of the other, yet retaining the system integrity provided by separate buses. When used in this way, the two interface sets will temporarily synchronize the two normally asynchronous processor buses so that information may be transferred.

If the F100 family is considered as a processing unit, consisting of an F100 with one or more interface sets and memory interface circuits, a non-multiplexed processor bus with in built direct memory access and bus arbitration circuitry, comprehensive fault detection, processor/processor and processor/special function processor abilities is the outcome. However, this system consists of at least five forty pin integrated circuits of high power consumption bipolar logic. The F100 was the first microprocessor family where the designers placed emphasis on high reliability, multi-processing capability. As such, solutions were found to problems that few other devices can cope with.

4.12 The VAX 11/780

As stated earlier, the VAX bus structure is discussed in this Chapter as, to date, the development of microcomputer bus structures has largely followed that of the minicomputer buses. The VAX is a recently introduced top end minicomputer from the Digital Equipment Corporation and any novel features of the VAX bus structure are a possibility for inclusion in future high performance microprocessors.

The major elements of a VAX 11/780 minicomputer are a microprogrammed

thirty two bit processor, one or two memory controllers, each of which can control up to a megabyte of memory, an adapter to enable the use of peripherals designed for the earlier PDP11 range (Unibus adapter) and a bus interface for newer high performance peripherals (the Massbus adapter). All these elements are linked by a bus of upto three metres length known as the synchronous backplane interconnect or SBI. It is, therefore, the SBI that will be examined for novel features.

The major innovation of the SBI designers is the recognition that a read bus cycle consisting of a combined "broadcast address and wait for response" will actually consume considerable bus bandwidth waiting for the retrieval of the data from memory or input/output devices. In a large system with megabyte dynamic memory and full error detection and correction logic on that memory, access times can exceed 500 nanoseconds, during which time the bus is effectively idle.

Provision is therefore made for transmitting an address and with it the identifier of the bus master requesting the transfer (processor, direct memory access controller, etcetera) along with the type of transfer required (read, write, double word read or write). Once this information has been broadcast, all devices will check the bus to see if the address corresponds to a location within that device. If this is the case, two cycles later, a confirmation code will be broadcast that indicates either i) no device recognizes the code (no reply), ii) device recognizes and will process command, iii) device recognizes and would process the command, but is currently busy with other tasks or iv) device recognizes but command is not valid for that device (for example, trying to write to a read only memory).

If the command is accepted, it will not necessarily be processed immediately, for example the memory subsystem can 'stack' several requests, and stacked requests will be processed first. At some future time, the

participating device will gain control of the bus and place the requested data on the bus, along with the identifier of the requesting device. This bus cycle is also acknowledged, with the requester indicating either that the data was received correctly, or with correctable errors, or a retransmission is required. Due to this "I'll ring you back with the answer" mechanism, it is possible for up to thirty two data transfers to be in progress simultaneously on the SBI.

The SBI is basically a thirty two bit multiplexed address/data bus. There is no overall bus master for the bus, all participating devices indicate their requirement for a bus cycle at the start of a bus free period by asserting their particular arbitration line. There are sixteen of these lines, of which the highest priority is used by all devices to retain control of the bus if additional cycles are necessary. It is, therefore, possible for fifteen devices to occupy the SBI, although the current maximum configuration for the VAX is eight devices. The lowest priority device is the central processing unit, then up to four Massbus adapters, followed by the Unibus adapter whilst the highest priority is given to the two memory controllers. As the memory controllers will be the most heavily used devices, and, as they cannot initiate bus transfers, merely obtain bus cycles to reply to requests, the controllers must be the highest priority devices. Failure to do this would lead to the SBI being flooded with requests to the memory subsystems which would fail because the memory sub-systems were still trying to gain a bus cycle in which to transfer data previously requested. Similarly, the low data rate peripherals (attached via the Unibus adapter) also require protection in this fashion to prevent their being locked out by the higher performance Massbus devices. The Massbus devices do not require this protection as the Unibus devices cannot transfer data fast enough to use all available bus bandwidth. Finally, the central processing unit is accorded lowest priority as

it initiates all the above transfers under software control. Most software tasks will have to wait for completion of the input/output transfers and so the processor should always defer to the devices trying to complete those transfers.

At the start of a bus transaction, all devices will recognize the right to the bus of the device asserting the highest priority arbitration line. If this device is a controller, it will place on the bus a twenty eight bit address and a four bit function code that identifies the action to be taken by the participating device, with a mask identifying the participating bytes at that address (each address contains thirty two bits), and a tag field indicating that the data highway contains address and function data and the identity of the device requesting this transfer. Once this cycle terminates, the SBI is again available for rearbitration. The confirmation lines of the SBI are always two bus cycles out of phase, thus giving devices time to test their ability to process the request. Therefore, two cycles later, the request will either be acknowledged or rejected. At some future time, the participating device will win the bus by arbitration and broadcast thirty two bits of data, a tag field identifying the bus as containing data. The ID field holds the identifier of the device that originally requested the transfer, whilst the mask field indicates that the data presented is being sent for the first time, or is a retransmission of data previously sent but corrupted in transmission.

In the case where the originating device is writing, the highest arbitration line is asserted to retain the bus for additional cycles after the address is broadcast with the write function code and, during these additional cycles, the data will be presented. Two cycles after the address, the confirmation of address reception will occur, and in the following cycle, confirmation of correct or incorrect data reception will be given.

This scheme is extremely complex, since all data transfers occur between devices that are equally responsible for arbitration and generation of control signals and, in this scheme, the processor is the lowest priority device. The nearest approach to this bus structure is found with the F100, where the participating device drives as many bus control signals as the processor, and a memory controller is used to interface to a memory sub-system. Again, in the F100 structure, the responsibility for both direct memory access and bus arbitration is shared by all interface sets rather than being vested in one direct memory access controller. As more microprocessor families provide intelligent peripheral controllers, such as the Intel 8089 or Motorola 68121, each of which has an on board processor, the use of distributed arbitration and direct memory access is likely to become more widespread. However, the use of separated address and data transfers is unlikely to occur as long as the technology used in the processor and the memory device is of a similar speed. Should either the memory or the processor become an order of magnitude different in speed, the ability to either break memory into separate units that can be interleaved by stacking requests with controllers, or provide direct memory access without interfering with slow processor accesses, will become desirable.

One further point of interest relating to the VAX is the method used for controlling the processor. The microcoded processor is controlled by an LSI 11 microprocessor which acts as a supervisor during the process of booting the VAX and also when fault diagnosis is required. Thus the engineer who services a VAX 11/780 uses a supervisory microprocessor to examine and alter the internal registers of the processor, as well as accessing any devices on the SBI.

4.13 Bus Structures : Summary

The development of microprocessor bus structures from the inception of

the device twelve years ago has broadly followed a manufacturer independent trend. The earliest, four bit microprocessors were designed to operate as complete units consisting of microprocessor, memory and input/output devices that were specifically designed for use together. As such, the bus structure could take any form that was implementable and could, if necessary, rely on the precise timing characteristics of the memory and input/output devices to ensure the correct functioning of the complete system. With the introduction of eight bit microprocessors, the increased flexibility of memory mixture and packaging required by the end users of the microprocessor devices, coupled with the acceleration of development in memory technology and the resultant need for industry wide second sourcing agreements, brought about a separation between the designers of the microprocessor, and those designing the memory devices it would use, who were now often from different companies. Although the timings of the memory devices have many similarities, the bus structures used to drive these devices show marked differences dependent on the particular philosophy of the manufacturers. The eight bit microprocessors can therefore be seen as processors and input/output devices from a particular manufacturer that are designed to interface to a wide range of unintelligent memory devices. For most of the eight bit microprocessors, little design effort was directed at allowing the connection of several processors into one system, the exceptions being the SC/MP and the M6809. Direct memory access devices for eight bit devices are usually amongst the support devices provided with the processor family, and therefore conform to the timing requirements of that processor.

The second generation sixteen bit microprocessors all possess schemes for allowing the use of multiple processors on the bus, although the comprehensiveness of the schemes varies widely. As the level of integration present within the processor has increased, the circuit complexity to provide

a logical and consistent bus interface represents less of an overhead. Particular exceptions to this overall trend of increased simplicity of bus timings are the F100 and the iAPX432 from Ferranti and Intel respectively.

The F100 uses a complex bus structure that offers high reliability, but compensates for this complexity by the provision of support devices that ease the interface to non-Ferranti products. The iAPX432 is the Intel microprocessor that uses object addressing and capability protection to ease the use of multiple, task sharing microprocessors. As such it has an extremely complex inter-processor communication protocol that uses objects (data structures) held in memory that the processors reference and manipulate with hardware. An interface to such a device would be extremely complex, but a support device (the iAPX43203 interface processor) is available that completely supports the iAPX432 bus structure, but which also allows external devices to manipulate objects, or by-pass the object addressing rules and treat memory as a linear array. The interface presented to external devices by the iAPX43203 is similar to that of the Intel 8086.

For new microprocessor products, the ability to utilize established families of memory and support devices is likely to act as a stabilizing force. If the expected benefits of a radically different bus structure are such as to force the production of devices using that bus structure, provision of a single interface device to enable the use of other peripheral support families will enable manufacturers to provide rapid support for the new product in the short term and will thus be desirable.

The viability of the proposed technique therefore rests with the older eight bit microprocessors, where provision of memory access by other intelligent bus masters was not a design priority. As stated earlier in this chapter, the fundamentally important signals are the address and data validators and the signals used for data direction control. An examination of

the timing diagrams presented shows that there is little fundamental difference between the various microprocessor families as, at the eight bit stage, the only bus devices expected were passive. Had the later concept of co-processors, now used with sixteen bit devices, been used with the bus structures found amongst the eight bit processors, the timing constraints imposed upon external, non-family, intelligent bus controllers would have been far more severe than is the case. However, the simplicity of the immature products enables the design of a processor to processor interface to be both feasible and economically valid.

5. The Memory Manager

5.1 Introduction

Given that two processors are to be connected so that the complete address space of the slave processor is available for examination by the software of the master processor, there is an obvious requirement for some scheme that enables the master processor to translate the address generated by the normal functioning of the microprocessor chip so that the problem of duplicate addresses can be overcome. As mentioned previously, the simplest example of this problem can be seen if two processors of the same type are used for both master and slave processors. Should the master processor wish to examine the reset location of the slave processor and the master places the address of the reset location on the bus, the memory of the master processor will respond. To overcome this problem, a translation mechanism must be placed between the master and slave processors that will take an address generated by the master (not corresponding to any master memory devices) and produce the required address to be fed onto the slave bus when an access is made.

As in any reasonably complex system, the number of free locations will be small, it is reasonable that the translation scheme be able to generate a signal that indicates which processors' memory is of interest, thus enabling any location to be used for slave access. Similar schemes have long been used by minicomputer and mainframe manufacturers as a method of extending the power of their machines as the falling prices of memory devices have made it economic to provide each task currently executing within the computer with its own memory space. Generally known as memory managers, these devices usually appear as a peripheral to the processor and have the advantage that no architectural changes are required within the processor to enable it to use the extra memory effectively.

Where a memory manager is being used to provide access to another

processor, there are several possible approaches that may be taken. These will be discussed in order of increasing complexity and both the organization as seen by the host processor and the necessary circuitry will be discussed.

5.2 Fixed window memory managers

The simplest memory management scheme possible is to restrict the ability of the master to access the slave. Here the memory map of the master has a permanent gateway to the slave which can only access as much of the slave's memory map as has been removed from the master's memory map. For example, if the master only has memory from addresses 0000_{16} to $7FFF_{16}$, the top line of the address bus can be used to generate the slave access request. Addresses 8000_{16} to $FFFF_{16}$, when generated by the master would be translated by hardware on the interface card. So, for example, if the slave processor was a Motorola 6800, the reset vectors ($FFF8_{16}$ to $FFFF_{16}$) and the direct addressing space (0000_{16} to $007F_{16}$) would both be of interest. It could be arranged that master addresses 8000_{16} to $BFFF_{16}$ would access slave addresses 0000_{16} to $3FFF_{16}$, whilst master addresses $C000_{16}$ to $FFFF_{16}$ would access slave addresses $C000_{16}$ to $FFFF_{16}$. Such a system is shown in Figure 5.1, whilst a possible circuit implementation is shown in Figure 5.2. This approach has many disadvantages and the most obvious is that, if the slave is a device with a far greater address space than the master (a Motorola 68000 with sixteen megabytes of address space), the master is effectively unable to see the memory map of the slave! Another problem is that the translation would change from slave to slave, reflecting differing areas of interest and this would lead to confusion as to the method of accessing a given slave address.

5.3 Single Window Memory Manager

In this case, the master processor has a fixed memory map. As usual, the memory decoding is performed to allow the generation of the necessary chip select signals. For one such signal, there is no corresponding memory.

Instead, the request by the processor for an access to this non-existent memory device causes the generation of a WAIT signal and, at the same time, signals are sent to the slave processor requesting the use of its bus. Once access has been granted to the slave, the WAIT request on the master is removed and the data and address buffers between the master and slave buses are opened. Those address lines generated by the master but not used in the chip selection process are passed through unaltered, whilst those that were used to generate the 'access select' signal play no further part in the access to the slave and their place is taken by signals generated by a latch. By altering the contents of the latch, the window can be pointed to any section of the slave address map desired. This is shown in Figure 5. 3 for two possible contents of the latch, whilst a possible circuit is given in Figure 5. 4. Note that if the latch generates more signals than were absorbed by the address selection process, the number of available address lines for transmission to the slave has effectively been increased. In this way, a Z80 which has sixteen address lines, can access any location in a Motorola 68000, which has twenty-four address lines.

At this point, the size of the window must be determined. Just as there are different sizes of memory chips available, so the size of the window can be varied by altering the number of address lines used in the chip selection generation. The trade off is between the number of times the access latch must be reloaded and the amount of space removed from the master address map. For example, if the window were one location wide, between each access, the access latch would have to be reloaded. This is obviously time consuming and effectively corresponds to the parallel port scheme of access described earlier. If the window is 256 locations wide, that number of successive memory locations can be accessed between altering the contents of the latch. However, if memory is not being examined, but rather the flow of program execution is

being followed and the program consists of a jump about a page boundary, the access latch could still need to be altered prior to each access. This is illustrated in Figure 5.5. To overcome this problem, the window should be as large as possible consistent with leaving enough working memory within the master processor.

Another possible cause of difficulty is when the slave processor makes frequent references to a variable area, again the translation latch will require frequent alteration. One possible solution to this problem is the provision of two or more windows, each independently selecting an area of memory.

5.4 Multi-window Memory Manager

As suggested above, the use of several movable windows provides extra flexibility. The principle is similar for two or more windows and the two window case will be discussed. Here two, preferably adjacent, areas of the master's map are used as windows and there are two access latches. When an access to an area occurs, the appropriate latch is used to generate the final address. Now one window can be used to access slave program data areas, while the other window can be used to access the slave program itself. Note that this does not solve the problem of the awkward program loop mentioned above, or the question of window size. Indeed either the allocated area of each window must be halved or the total window allocation doubled.

5.5 Integrated Host/Slave Memory Management

Taken to the logical conclusion, the window system will have *all* chip selects accessing a mobile window that points at memory, some that is available to both master and slave, some that is available to the master only. This is the form of memory management schemes as used on mainframe and minicomputers. Diagrammatically it is shown in Figure 5.6. The memory address eventually generated will be described as the *physical* address, whilst that

generated by the processor will be known as the *virtual* address. The use of this scheme solves several problems. Several windows can be assigned to memory controlled by the target without suffering a loss of working memory since they can be reassigned to access host memory when not in use for debug purposes. Also, it becomes possible to assign any window for use, and a window not currently in use for other tasks can be chosen for the purposes of accessing the slave. In the same way that the effective address space of the host processor can be increased, so that it can access the entire address space of a microcomputer with more address lines, so the memory that is only attached to the host can be expanded. This enables the system designer to place in read only memory many items which are infrequently used, without the attendant problem of using up valuable address map space.

The problem of correct window size determination is not solved by this technique and the trade off remains that, as the window size is decreased, better control of memory disposition is obtained. It will, however, take longer to alter the memory map when such changes are required.

In fact, the problem of the window size is more restricted by the limited range of low cost, high speed TTL compatible random access memory chips, and a window size of four kilobytes per page was eventually adopted.

5.6 Variable Page Size Memory Management

Since this work was originally undertaken, the falling cost of semiconductor memory coupled with the increasing power of microprocessors has encouraged the production of a range of memory management chips that are usually dedicated to the support of a particular processor. In general these devices are unsuitable for the system under discussion due to the processor specific nature of their design. In one respect, the problem of the window size trade off has been resolved by specifying two fields for each memory page. The first generates an offset to be added to the processor produced

address and, in this respect, differs only slightly from the methods already discussed. However the second field specifies the length of the window, usually as a binary weighted size, that is, the basic window size element can be repeatedly doubled in size to the next largest above the desired size. In this way, the user can specify up to thirty-two windows, each of which precisely matches the necessary configuration for the task that inhabits that window. Note that the primary purpose of such chips is not the expansion of a limited address space, as the device described (the 68000 memory management unit) can only be used with a processor that can directly address sixteen megabytes, rather they are to afford multiprocess protection. As used, each process can only access its own working and program space. Should an adjacent section of physical memory contain elements of another process, there is no danger of accidental access, as the memory of the next process is not currently reachable and no translation can be performed to generate that address. The task can only change the current map by requesting supervisor intervention as the execution of a supervisor call alters the memory map, but also returns processor control to the operating system, which is the only program that may operate on the memory manager itself.

5.7 The New Memory Management Scheme

If the previously suggested scheme of sixteen windows, each of four kilobytes, is accepted, it then becomes necessary to decide the number of address lines that will be provided by the memory manager for the processor andhow these will be organized amongst the possible memory locations, i.e. host memory, slave memory etcetera.

Most microprocessors still feature an address bus of sixteen lines or fewer, whilst the next common increment is to twenty lines. It is therefore necessary that the memory management should be capable of generating at least another sixty-four kilobytes of map.

The most common processors met in the development and educational environments are the eight bit processors. This is largely because the four bit processors are usually used for the extremely high volume markets, where the difficulty of task development is more than offset by the cost savings in production. The sixteen bit processors are used only in two cases. The first use is as a minicomputer replacement, that is the system is developed and then used as a general purpose computer. Such systems are software intensive and as such require high level language support rather than hardware development features. As such, the developmental support that is needed at the hardware level is a short term requirement only. The second instance where the sixteen bit processors are used is in embedded computer systems where the task is too complex for one or several eight bit processors. The user is therefore forced to use the more expensive, more powerful processors which are generally the sixteen bit devices.

Because of this, the design of the memory manager is subject to yet another set of conflicting requirements. To provide a general purpose facility it should enable the host processor to access any memory location anywhere in the sixteen bit processors' address space, which will increase the cost and complexity of a system that will primarily be used with smaller processors which do not require a large, complex memory management scheme. The minimal requirement is that the memory manager provides another sixty four kilobytes of memory space that can be used to examine the entire memory map of the common eight bit processors. The maximum requirement is that it should be able to cope with all current and foreseeable processors. The first case is extremely limiting, whilst the second case increases the complexity of the system to an unworkable extent.

The method used to overcome this dilemma is to split the memory manager into two sections. The first section will be resident within the host system

and will provide enough support for the most common processors as well as any address bus extension found to be desirable within the host system itself. As the interface card changes with the type of processor to be used as the target, it becomes sensible to place the memory management extension on the interface card, where the additional complexity and cost will only occur when necessary.

Whilst this resolves the two conflicting design goals, the partition of the memory manager into two units requires that a decision is made as to the size to be used for the minimal memory manager. In practice, this is influenced by the high speed memory devices that are used to implement the memory manager and as with the page boundary decision, the ready availability of four bit wide sixteen location TTL compatible devices leads to the adoption of an address bus designed around a formula of (host processor lines)+4*n, where n is the number of chips to be used in the memory manager for the purposes of bus extension.

In fact the address bus of the host was extended to twenty bits in this exercise, as this also left the board design within the bounds of a second constraint, that the number of available bus connector pins was not exceeded! This leaves an address map as shown in Figure 5. 7. As can be seen, the normal Z80 map can be visualised as existing as the bottom page in the megabyte area. This is the only sensible location as it is here that the extended high order address lines are at zero, and are thus numerically compatible with those normally generated by the processor. The next factor to be considered in the design of the memory manager is the precise method of identifying which accesses are to be transferred to slave memory, and which are to be handled by memory resident within the master system. There are two possible methods of achieving this distinction. The first is to use one of the newly created high order address lines for the purpose, for example in a twenty bit address bus

(A0 to A19), A19 could be used to indicate that the slave memory is to be accessed. In other words, any address between 00000 and $7FFFF_{16}$ would be treated as being a master memory access, whilst any access between 80000 and $FFFFF_{16}$ would be seen as a reference to slave memory addresses 00000 to $7FFFF_{16}$ respectively. Such a system would be quite acceptable but for the requirement that the memory manager be extendible. If such a technique were used, the transfer to an extended memory manager would imply either that all slave addresses were considered to have an in built 80000_{16} offset, or that the extended portion of the memory manager would alter the action of the top bit. As the memory manager is there to remove any difficulties in accessing the slave memory, both the above suggestions are obviously unacceptable.

The second method of differentiating between the two memory areas is to assign to all memory in the system two sets of descriptors. The first is the address of the memory placed in the largest address space present, (i.e. location 0100_{16} would be known as 00100_{16} in a megabyte addressing space) and the second descriptor specifies which location is the 'home' of the memory (i.e. master bus, slave bus etcetera). The second suggested descriptor has become known as the attribute of the memory, and by logical extension has also been used to overcome several other problems in the design of the system.

As currently implemented, each of the sixteen virtual pages has an attributes field which can be used to specify where the memory is resident, whether it can operate at full host speed, or whether it should have additional clock cycles inserted when accessed for either the instruction fetch cycle (which on the Z80 is shorter than a normal memory reference) or whether such cycles are required for any memory reference. Another useful feature of the attribute field is the provision for protecting the memory being accessed from write cycles. If such protection is required, the memory access will not take place and this feature is selectable on two kilobyte

boundaries rather than four. Then, one page may serve as a message passing area for two processors, each of which only have access to one half of the memory area for writing (usually each area will only be writeable by one processor) but can read either half. From this it is apparent that two attribute fields are required, one for the master accesses and one for the slave. This is indeed the case and the high order address line random access memory is also repeated for the slave. This gives the ability to control slave accesses into the master.

The memory management scheme is the single largest addition to the host system to adapt it for the multi-processor task. It is, therefore, necessary that the form and location of the memory manager will decide the location of all other components of the system. The logical position for the memory management circuitry is alongside the supervisory microprocessor. Here, all the address lines of the processor are present and the control signals are available with minimal delays. This is important, since no other address decode circuitry may begin processing the information held on the address lines until the completion of address translation by the memory manager. The delay introduced by two sets of 74LS245 buffers (bidirectional so that other devices may drive the address bus towards the processor during direct memory access), one driving off the processor card with another driving onto the card containing the memory manager, is typically 16 nanoseconds with a worst case timing of 24 nanoseconds. As all other memory devices, except those on the processor card, would perceive an identical delay in the control signals, the effect of this delay will be minimal. Other considerations that will have more influence on the siting of the memory manager include the number of signals that must appear to emanate from the processor when, in fact, the memory manager is generating them. If a large number of such signals exist, the number of dedicated lines between the processor and the memory manager will

exceed the capacity of edge connectors, etcetera.

5.8 Memory Manager Specification

The insertion of circuitry that can be used to modify the address lines of a processor enables, with little extra complexity, the addition of other facilities which ease the task of programming a computer system. Therefore, the description of the memory manager will include facilities which are not directly relevant to the establishment of a processor to processor interface, but which share the same circuit components in the finished circuit design.

When power is first applied to the supervisory microprocessor system, the contents of the bipolar random access memories that make up the translation section of the memory management unit are undefined. Therefore, if the memory management unit was acting as an address translator at this time, the contents of the memory map could not be determined. This implies that when the processor starts execution at location 0000_{16}, the memory device that would normally respond to this location would almost certainly not be selected and control of the processor under such circumstances would be lost. As the function of the reset signal of the processor is to return the system to a known state, it follows that a similar action must be taken by the memory management circuitry on receipt of a reset. An important consideration is, therefore, that the memory management unit must be removed from the circuit by a reset, whether caused by power on or the operation of a reset switch. At the same time, the contents of the memory manager will, if the reset is not due to power on, be both valid and unchanged. Action to disable the memory management unit should not therefore alter the contents of the bipolar random access memory. In addition, the removal of the memory management circuit, which will normally be responsible for driving the added address lines A16 to A19, will leave these lines in an undefined state, as they are not generated by the processor. Circuitry must be provided that forces them to a known state after a

reset and the most logical value is 0, as this forces the generation of twenty address lines which are numerically compatible with those generated by the processor.

An important feature that can be added to the memory management is the provision for an offset to be added to the reset address of the microprocessor (0000 for the Z80). This enables several different reset programs to be placed in different EPROMs within the system and, by using switch selection, the desired program will be executed on a processor reset. For example, in the EPROM resident at 0000, a monitor program that requires only the terminal to function correctly and allows examination of memory and input/output addresses is common. Such a monitor will have provision for executing programs at other locations. However, if a system is permanently equipped with disk drives, a program that automatically attempts to boot the disk operating system is preferable in that it removes the need for inexperienced users of the machine to remember another, and vital, sequence of commands. The same effect can, of course be achieved by replacing the program at location 0000 with the new program that will automatically boot the disk, but, if a failure occurs on the disk system when examination of registers etcetera is required, the facilities of the monitor have been lost.

In this system, the reset offset takes the form of eight switches which are used to replace the high order eight bits of the processor generated address bus for the first instruction after a reset occurs. Hence, when the processor issues the address 0000_{16}, the address seen by memory decode circuitry is $XX00_{16}$, where XX is the value generated by the eight switches. Implementation of such a facility is readily achieved on the Z80 by an examination of the $\overline{M1}$ line. After a reset, the value presented by the switches is used for the top eight address lines until the beginning of the second $\overline{M1}$ cycle, which marks the start of the second instruction after reset. At this

point, the normal address lines are allowed to operate.

If the first instruction in an EPROM at address $D000_{16}$ is "jump to location $D003_{16}$", with that instruction occupying locations $D000_{16}$ to $D002_{16}$ inclusive, the program counter will take on the value $D003_{16}$ before the processor address lines are re-asserted and the processor will therefore continue to execute under the control of the $D000_{16}$ EPROM. Whilst this technique removes the requirement that the first instruction be a particular number of cycles, the first instruction should not be of the form "load register from extended address" if the address lies outside the page the switches point to. This is because, until the start of the second instruction, *all* read accesses, including data fetches, will cause the top address lines to be replaced with the value of the switch. In practice, this is only likely to occur at location 0000, since elsewhere, the first instruction must be a jump to enable the retention of processor control.

The final addition that can be included in the memory management unit design is the provision for placing a known value on the high order address lines during input/output transfers. As the input/output address space is only 256 bytes long, the top address lines are undefined during a data transfer. If the memory management circuitry is used to set these lines to a known value (0), then there is a reduction in the number of address line transitions during input/output cycles, thereby reducing the system noise. More significantly, the use of a constant value ensures compatibility should a future design incorporate an input/output address space manager designed to increase the capability beyond the Z80 limit.

It can be seen that the contents of the high order address lines will be derived from four sources. These are i) for the first instruction after a reset, switches, ii) until the memory manager is initialized and activated, the processor bus and, for the extended lines a constant value of 0, iii) when

the memory management unit is on, the output of the translation bipolar random access memories. This is best achieved by the use of a four way multiplexer and a block diagram of this arrangement is shown in Figure 5.8.

5.9 Memory Manager Initialization

The ability to access the memory management unit and alter the translation RAM contents is of prime importance. The locations to which new data destined for the memory manager is written cannot be in the memory address space of the microprocessor, as it would then be possible to "lose" the memory manager itself, thus making further changes to the memory map impossible until the processor and memory manager were reset.

The up-date locations therefore lie in the input/output space of the Z80, which, as already stated consists of only 256 locations. As each of the sixteen pages into which the memory manager is divided require at least four bytes, the total input/output space requirement would be sixty four addresses, or a quarter of the entire space available. Whilst this is feasible, the desire to allow memory management unit extension to take place would require two or three more bytes for each page. If the memory management unit is to be consistent, irrespective of extensions, these locations would have to be interleaved with the input/output addresses already mentioned. To reduce the demand on the input/output space and yet to allow the logical extension of the memory manager, provision was made in the input/output space for one page only, the "current" page. The current page , which is available for up-dating, is selected by writing the required page number to another input/output location.

A further requirement, largely dictated by programming experience, was that all parts of the memory management unit should be readable as well as writeable. If this is not the case, copies of the memory management unit contents would have to be kept in random access memory accessible to programs

and failure to keep the copies up to date could result in loss of processor control. A further complication, if this method is used, is that the memory in which the copies are stored is also a candidate for removal from the memory map. Similarly, interrupts that occur during execution of a memory manager up-date might require the ability to alter the map temporarily and the programmer of the interrupt routine might not know the location of the tables held in another task. The ability to read the current contents of the memory management unit ensure that the information is accurate, yet involves the provision of three more data buffers and their associated decoding logic.

The actions required to up-date the memory management unit are therefore i) selection of the appropriate page by writing the page number (0 to F_{16}) to an input/output port (location 44_{16}), ii) examination of the values currently in the memory management unit, performed by reading locations 40_{16} to 43_{16} and up-dating as necessary (writing to those locations). This reduces the total number of input/output locations required for memory management from sixty four to five.

The address supplied to the bipolar random access memories would normally be that generated on the top four address lines of the processor (A12 to A15). During the up-dating of the memory management unit, the address of the appropriate bipolar RAM location is provided by port 44_{16} and hence provision must be made to apply the output of port 44_{16} to the address lines of the bipolar random access memories during the period of the up-dating cycle. In practice, as the translation section of the memory management unit is never used during input/output cycles, it is easier to apply the output of port 44_{16} to the address lines of the bipolar RAMs whenever the processor is engaged on an input/output cycle.

One further complexity, introduced by the bipolar nature of the memory used to produce the translation section, is that the timing signals generated

by the Z80 for data strobing are not suitable for use with the bipolar RAM. The write and input/output request lines are also used to control the data buffers on the edges of the circuit boards and, as the buffers are switched off, the bipolar RAMs, being members of the TTL Schottky family, are fast enough to accept the new data, usually garbage. A special write signal is therefore generated within the memory manager which is extremely short, and occurs in the middle of the normal Z80 output cycle when the data bus is stable throughout the system.

5.10 Provision for Direct Memory Access

As mentioned previously, the slave processor is able to access the memory of the master processor. To provide control over the accesses, the interface between the two processors can either inhibit or allow accesses from the slave to the master. To provide a further degree of control, and also to increase the possible access range of the slave if only a limited address space is available for assignment to master accesses, the memory management unit is also used to translate addresses emanating from the slave processor. The sixteen lines sent from the slave to the master have the upper four bits modified in the same fashion as those of the master, but using a different translation table ("slave access page address" rather than "master access page address"). In the present system, any accesses from the master to the slave are restricted to the bottom sixty four kilobytes of the Z80 address map. Hence the translation RAM for direct memory access is only four bits wide, not eight, and the top four lines (A16 to A19) of the extended bus are held at zero by the same circuitry that acts when the memory management unit is switched off.

In providing for direct memory access, the flow of address information becomes extremely convoluted. The problem occurs since the device performing direct memory access will place sixteen bits on the bus. Slave microprocessors

with more than sixteen lines perform an address decode with the upper lines, hence only sixteen will be available. The lower twelve bits are unmodified by the memory manager and hence can be sent through the system on the normal address lines by reversing the direction of the address buffers. The upper four lines, if placed on the bus in the locations used for the output of the memory management unit, would conflict with those values generated by the memory manager in response to the request for direct memory access translation. As a result, either the memory management unit output would not be broadcast with no translation taking place, or the memory management unit output will be broadcast and the value generated by the slave processor will be lost with the memory manager not having a valid address to translate.

It is, therefore, necessary to separate the incoming direct memory access address into two parts, the twelve bit address which is unmodified and hence can travel through the system on the normal address bus, and a four bit part which must travel down a reserved path. All boards receiving address information during direct memory access must correctly identify the source of the various sections of address, and enable address buffers accordingly.

The lines used to carry the incoming four bits of direct memory access address to the memory manager are already available and connected to the correct section of the memory manager circuitry. As already stated, the memory management circuitry is extendible and, for the extensions to the memory manager to operate correctly, they must receive the original, unmodified version of the Z80 address, so that these lines may serve as address lines into the additional bipolar translation memories. Provision must therefore be made for the original top four Z80 lines to be transmitted at all times. During direct memory access, these lines contain no useful information as the Z80 address bus will be floating. If the buffers used to drive these lines are bidirectional and reversed for direct memory access, an address placed on to

these lines will be propogated back through the system and be observed by the memory manager as a value placed there by the Z80. If the bipolar RAMs used to perform the translation are also selected according to the state of the bus acknowledge line $\overline{\text{BACK}}$, the memory manager will operate correctly.

6. Hardware Description

6.1 System Layout Considerations

As long as a microcomputer system occupies only one printed circuit board, the positioning of the components will be determined by convenience. Usually, input/output devices will be near to the relevant connectors and other circuit elements will tend to be organized as functional blocks, thus reducing the number of interconnecting traces that run extreme distances. The limitations on system capability are solely those imposed by cost and board size. At some point, the single printed circuit board becomes too large or too expensive to manufacture and a design consisting of several printed circuit boards will be the result. With such designs, a further complication appears, that of splitting the various functional blocks between the several boards. Functional blocks will not usually be split over more than one board as the number of connections between the elements of a block far exceeds that between blocks, so that, to break up such a circuit, attracts a severe penalty in terms of connector requirements. The design of the particular system to be described is too extensive for one printed circuit board of the desired format (double eurocard) and the basic Z80 unit requires two printed circuit boards. During the design, it was therefore necessary to identify the various functions within the system and allocate them such that both boards could be of the same size. Due regard also had to be given to any other constraints upon board function imposed to achieve other design goals.

The major functional blocks identified for this system are the processor and associated reset circuitry, the memory manager, random access memory, read only memory, input/output peripherals and the memory mapped visual display unit. A design constraint imposed was that the processor card should be able to run without the second card. This allows the use of the processor card as a stand alone Z80 based microcomputer for simple systems and it can execute test

programs to assist in the commissioning of the second and subsequent boards in the system. One important use of the single card is for the provision of terminal facilities, but for this task, the input/output and visual display unit sections must both be available to the processor. Similarly, the memory manager is not required in a simple system and it is also desirable for it to be near to the interface to shorten the path for addresses that are generated by the slave and need translation during direct memory access. This approach led to the processor card containing all input/output peripherals and the visual display unit, carrying, in addition, enough EPROM and random access memory to contain a monitor and allow the execution of small test programs. The ability to run simple test programs allows the generation of repetitive waveforms, which in turn enables the testing of other system components. That such a facility can be provided by the processor card alone has two advantages; only one card need be tested at once and the test program will continue to run even though other units have faults that force erroneous signals onto the backplane. If the processor is dependent on another card for the provision of program or data storage, an address or data line short circuited on the backplane, or a bus driver functioning incorrectly, will cause the test program to fail. The system that resulted from these aims is shown in block diagram form in Figure 6.1.

All boards in the system have been designed so that one eurocard connector is completely occupied by the system bus, whilst the second is available for input/output. The exception to this scheme is in those cards with a bus to bus interface function (the second board and the interface boards). Here, one connector is used with one bus and the other provides a connection to the second bus. Such a layout allows all connections to peripheral devices to be wired into the rack, so that boards are removable without the danger of damaging ribbon cable connections.

In the following sections, each functional block, its position and relationship to other blocks will be described.

6.2 The Processor

The presence of the processor identifies the central board of a system and thus has led to the use of board 1 to identify the card containing the processor. The processor used with this design is the Z80A, which is capable of operating with a four megahertz clock. The reset circuitry of the Z80 can vary from a simple debounced switch to the circuit employed in this system which offers several facilities.

When used with dynamic random access memory, a simple reset switch is insufficient as the processor, if reset at certain stages of execution, will after the release of the reset signal, perform an aborted bus cycle. Whilst this will not cause electrical damage to the system and will not affect the contents of static random access memory, it is possible that the shortened bus cycle will allow dynamic random access memories to read data into internal buffers (which destroys the data in the memory storage cell itself) without allowing the rewriting of data (performed automatically by logic internal to the memories). It is therefore desirable, if the destruction of dynamic RAM contents is to be avoided, to synchronize the generation of the reset signal with a processor output that only occurs on occasions that will not cause the aborted cycle, in this case, the instruction fetch signal ($\overline{M1}$). As the processor is responsible for the refreshing of dynamic memory, and this does not occur whilst the reset line is asserted, reset sequences must be of short duration. Therefore, the reset line and power on reset circuit are used to trigger a monostable, with the input to the monostable clocked by the $\overline{M1}$ signal. This arrangement ensures that the contents of dynamic random access memories will be preserved through a system reset. A falling of this circuit, which cannot be resolved in a simple manner, is that if an interface locks up

and generates an indefinite wait signal to the Z80, a reset cannot occur as the $\overline{M1}$ signal is also absent. This state can only be left by removing the offending board, or by removing power from the entire system. The latter alternative is not as severe as might at first appear, as the wait signal will stop the processor from refreshing the dynamic random access memory, which will therefore contain corrupt data. A method of overcoming this problem is available in that a wait time-out monostable could be used to prevent the wait line being active for long enough to cause corruption of the dynamic random access memory. For example a wait time of one millisecond is acceptable, such a time being adequate for all normal memory accesses and yet still well within the refresh requirements of all dynamic memories. A drawback to the above system is that the program that requested the data, thus initiating the wait state, would be allowed to continue with a random data value obtained from bus capacitance and false results could be obtained.

A further feature of the reset circuitry relates to the resetting of Z80 peripheral devices. The interrupt daisy chain of the Z80 demands that peripheral devices monitor the processor instruction fetch signal ($\overline{M1}$), as well as controlling the two daisy chain lines (IEI, IEO). In general, this does not allow the reception by such devices of the reset signal. All Z80 peripheral devices have an internal power-on reset which is used to set the device to a known state. However, should it be necessary to re-enter this state during operation, an alternative means of forcing the device to reset must be used. The method employed by Zilog is to use an otherwise impossible combination of processor control signals, all of which are already received by the input/output devices concerned. The $\overline{M1}$ signal is used to indicate the fetching of an instruction. As such, it will be used in conjunction with the read signal and, for normal cycles the memory request signal ($\overline{MREQ}$), or the input/output request ($\overline{IORQ}$) if the cycle is an interrupt acknowledge. Should

the $\overline{M1}$ signal become active without the read signal, Z80 peripheral devices will enter the reset state. Therefore, the reset circuit in use has two keys, one to cause the processor to be reset, and a second, acting only in conjunction with the first, causing both processor and input/output devices to be reset.

An important indication provided by the processor is whether a HALT instruction has been executed. When this occurs, the halt line becomes active and, in this system, the line is used to control a red LED on the printed circuit board, which is repeated on the keyboard of the unit.

A further facility offered is the generation of a non-maskable interrupt for program single stepping purposes. If a non-maskable interrupt occurs during an instruction, the instruction will be completed (or in the case of block instructions the current iteration will be completed) and then control will be passed to a routine at location 0066_{16}. A circuit provided can be triggered by applying a signal to an input/output port, where upon it will count three instruction fetch signals before triggering a non-maskable interrupt. This allows the reloading of the accumulator (which would be holding the data sent to the input/output port), the execution of the return from interrupt instruction and the first instruction of the main program, at which point the interrupt is generated. The implementation of this feature is such that programs in read only memory can be stepped through, a technique impossible with software only single stepping techniques. If software is provided to cope with the display of registers on the reception of a non-maskable interrupt, this technique also provides a convenient method of regaining control of a program that has entered an infinite loop. Therefore three methods of non-maskable interrupt generation are catered for, the single step circuit, the front panel switch and, finally, any other device attached to the $\overline{NMI}$ line which is fed throughout the system. Provision is also made in

software for the redirection of non-maskable interrupts to routines specified by the user, a technique required if option (iii) above is to be used.

6.3 The Visual Display Unit

This circuitry is also contained on the processor card, thus allowing the creation of a single card terminal. The display format is based upon that of the first prototype, a Nascom 1. This format was used for three main reasons. When the first card was designed, the video interface devices were relatively expensive, and little experience had been gained of such devices. The adoption of a screen format compatible with that of a similar Z80 based machine allowed the running of software that was already available. In particular the monitor program, BASIC language and a combined assembler/editor were all available and expected the Nascom 1 screen format.

The format is forty eight columns by sixteen lines and can, because of the low bandwidth, be displayed on a domestic television. This has allowed the provision of more screens at costs considerably lower than systems that require specialized video monitors. Limitations of the system brought about by the adoption of this screen format have proved to be the difficulty of using software designed for use with eighty column terminals (such as software supplied with CP/M) and the high cost, few facilities and large size of the circuit compared to modern video generator based circuits.

The basic timing of the circuit is provided by a sixteen megahertz crystal oscillator that is also responsible for the generation of the processor clock. The output of the oscillator is fed into a divider which generates 8, 4, 2 and 1 megahertz and these frequencies are passed to a four to one multiplexer. Two switches are used to control the multiplexer, thus allowing any one of these signals to be fed to the system clock. The eight megahertz signal is not frequently used, due to the marked reluctance of the Z80A to operate at such a speed. However, newer processor versions will be

able to use this clock frequency. The eight and one megahertz signals are fed to the video timing circuit as dot and character clocks respectively, via a monostable circuit that reduces the pulse width to a minimum. These signals are used to control the loading and clocking of an eight bit shift register connected to the output of a character generator read only memory. An equal mark/space one megahertz signal is fed into the video timing chain, composed of 74LS163 synchronous counters. This chain generates, by appropriate combination of outputs, the screen RAM address corresponding to a given screen location, line and frame blanking and sync pulses and a signal used to modulate data being outputted to the audio cassette interface.

The line and frame sync pulses are also available to software via an input port, thus allowing, firstly, the synchronization of processor accesses so that no disturbance is seen on screen during up-date operations and, secondly, the provision of processor speed independent timing pulses at rates of one pulse per sixty four microseconds and one pulse per twenty milliseconds. The random access memory address signals from the timing chain are fed to the memory through a set of 74LS157 address multiplexers, whilst the other input to these multiplexers is from the processor address bus. During normal operation, the video timing chain addresses select the appropriate memory location. Whenever the processor selects the video memory, the video timing chain is disconnected until the processor has finished. If this occurs whilst the electron beam is on screen, a characteristic "snow" appears, as the character generator ROM inputs will acquire the processor up-date value, which will be displayed in the incorrect place. As the output from the video RAM (which is separated from the main processor data bus by a buffer) must be latched, provision has been made to reduce the snow by clearing the latch during accesses by the processor. As this would select the character corresponding to 0 (an empty rectangle), the latch is both preceeded

and followed by an inverter on bit five. Thus, when the latch iscleared, the video character generator is sent the value 20_{16}, which is a blank space. Normal input to the character generator is inverted twice and is unaffected. The use of this technique offers significant improvements over other circuits in more common use, such as a monostable that forces the blanking output low whenever the processor acquires the video RAM. Still further improvements can be obtained by using the line and frame sync pulses to control accesses to the video screen. Such circuitry restricts the access by the processor to non-display periods and limits the up-date rate of the screen and was therefore not built into the hardware, but left as an optional software feature. The ASCII character set uses only seven bits of the eight available in the screen memory. The top bit is used to select a simple block graphics character set that can be used for the generation of diagrams. Each character position of the screen is divided into four areas, and each area can be switched on or off independently of the others. In addition, the colour of the area is selectable, the same colour being given to all illuminated areas in one character position. Those areas not illuminated take on the background colour of the screen, which is also selectable. Areas which are selected to be the same colour as the background exist, but are not visible, thus allowing a primitive conceal/reveal by manipulation of the background colour. Eight colours are available, being all possible combinations of red, green and blue, when output through the UHF modulator or as a composite video signal, the colours are seen as different levels of brightness.

Further screen facilities are the provision of black on white or white on black text, selectable by switch and the injection of the background colour into letters, allowing software selection of text colour, albeit on a screen wide basis. The output from the video circuit is available in three forms, composite video, UHF (provided by a modulator on the back panel) and RGB drive

for colour monitors.

6.4 Input/Output

The first board holds all input/output devices requiring connection to the outside world. The basic elements of the input/output section are two Z80 peripheral input/output (PIO) devices and two National Semiconductor INS8250 universal asynchronous receiver/transmitters (UARTs). In each case one device is devoted primarily to system use, whilst the other is left free for any task required by the user. The user PIO is connected directly, via the second eurocard edge connector, to a twenty five way D range socket on the back of the case. System software does not refer to, nor will it set up, this PIO. The system PIO is used to latch in data from an ASCII encoded keyboard. To achieve this, one half of the PIO is operated in strobed handshake mode. The signal produced by the keyboard on every key depression is used to latch data into the PIO and the same signal is also fed into the other half of the PIO as a simple unlatched input which can be used to time the period for which the key is depressed. This facility allows the provision of auto repeat for interactive screen based programs. The data ready signal of the latched section of the PIO produced in response to the keyboard strobe signal, is also fed into the unlatched section of the PIO to act as a new data ready signal which is automatically cleared once the data has been read from the input port. The data held by the keyboard port is always valid and represents the last key pressed. To allow the use of the widest possible range of keyboards, the incoming keyboard data ready signal can be inverted, if required, by switch selection. Of the eight bits provided by the unlatched section of the system PIO, two have already been described in relation to the keyboard input, a further two accept the frame and line sync pulses from the video section. The remaining four are outputs, of which three select the background colour of the screen and one is used to trigger a non-maskable interrupt for single

stepping.

The two UARTs share a common 1.8432 megahertz crystal oscillator. This frequency is convenient in that it can easily be divided down to provide, with the required accuracy, the common RS232C baud rates. The INS8250 has in built divide circuitry which is controlled by software and therefore the system offers software selection of transmission/reception rates. The user UART is provided with RS232C interface circuits on six lines : RX, TX, $\overline{RTS}$, $\overline{CTS}$, $\overline{DTR}$, $\overline{DSR}$. This allows the generation of all common communication protocols under the control of the processor. To provide assistance to the software responsible for maintaining the protocols, the INS8250 provides two versions of each line input signal. One bit is a direct copy of the state of the input signal, whilst a second bit indicates if the signal has changed state since the last examination by the processor.

As the UART provides a divide circuit for the generation of baud rates and, as this operates by using a dividing integer rather than the selection of one possible rate from a fixed range, the UART can be used, if not required for data transmission and reception, as an elementary real time clock. To achieve this, the baud rate output of the user UART is fed into the $\overline{CTS}$ input of the system UART, which is not available for communication protocol purposes. Thus, by selecting the appropriate divide rate, software can have both a copy of the square wave produced, and a flag indicating whether a change has occurred. The increments available are quite small, being given by $1.8432 \times 10^{6} / 65536 \times 16$ i.e. approximately 1.75 microseconds.

The system UART provides a simple RS232C input/output link (RX, TX only) or a modulated signal that can be fed to an audio cassette to allow data or program storage. Similarly, an amplifier and a demodulator is provided for data recovery. These facilities are selected automatically by the presentation of a signal from either source, such that if both inputs are used

simultaneously, data corruption will result. To provide additional flexibility, the UART and modulator/demodulator can be by passed completely with the cassette interface being driven by a UART control bit and the input similarly sensed. This allows full software control of format at the expense of increased complexity. Two UART output bits are used to drive two transistor circuits. The first drives two LEDs, one on the printed circuit board and the other on the front panel, that can be used to indicate that the cassette recorder should be switched on or off. As these LEDs are controlled solely by software they can, of course be used for any other purpose desired. The second transistor drives a speaker positioned behind the front panel which can, with suitable software, be used to generate sounds. The state of the non-maskable interrupt key on the front panel is also available via a bit input on the system UART. This signal is provided so that a non-maskable interrupt routine can determine whether the NMI was the result of a device requesting service, or a front panel generated abort.

All the above mentioned signals are connected via the second eurocard connector and the chassis wiring then transfers the signals to the appropriate connector on the back of the unit. The eurocard edge connector is fully occupied and there are several signals that cannot be taken off board. These signals, bit input/outputs from the two UARTs and the top bit of the keyboard PIO (the keyboard only generating seven bit ASCII), are collected onto an integrated circuit socket that can be used to feed in signals if required.

6.5 Board 1 : Miscellaneous Functions

Several sections of circuitry are included on board 1 to enable it to provide services that do not require the full capabilities of the relevant funtional block. For example, the board includes two kilobytes of EPROM and one kilobyte of random access memory that can be used to allow the board to execute monitor and test programs. Although the processor is also on board

one, the address lines A12 to A16 are not attached to the processor. Instead, with A16 to A19, the memory management unit is the source. Thus the processor lines A12 to A15 leave the board along lines designated NA12 to NA15 and feed only the memory management unit. The lines A12 to A19, returned by the memory manager, must all be low to select memory devices held on board, which thus corresponds to addresses 00000 to $00FFF_{16}$. To enable the operation of the card without board 2, these address lines are terminated on the edge of the card with pull down resistors. If the second card is not present, the four kilobyte address map of board 1 (two kilobytes EPROM, one kilobyte video memory, one kilobyte program RAM) is imaged throughout the sixty four kilobyte map of the Z80. When the second board is present, the memory management unit can remove the board 1 devices from the memory map. This is necessary if the CP/M disk operating system is used as this expects random access memory at address 0 upwards.

The use of the Nascom BASIC read only memory, whilst allowing the ready provision of the BASIC language, creates several problems. The program uses monitor subroutines for all input output, whether to keyboard/screen or cassette, except for two special cases. To provide the ability to halt a running BASIC program, the keyboard is tested before the execution of every line of BASIC. If the escape key is pressed, the program halts and control can be regained. The original Nascom keyboard was an unencoded device and relied upon the processor scanning the keyboard to detect key depressions. Due to the time taken by this method, the BASIC interpreter undertakes a simple test of the input port of the keyboard whilst the scan lines are left selecting the appropriate column. Therefore, a running BASIC program will not see any key pressed on the encoded keyboard used for this system. The input/output reset key is used to place the appropriate value onto the data bus whenever the key is depressed and no other device is driving the bus. The second exception

relates to the cassette storage of BASIC programs when the Interpreter attempts to write/read a special header block containing the program name. As the UART has been replaced with a more powerful type, these attempts also fail and the data must be stored and retrieved under monitor program control. In order to accommodate the BASIC ROM, no input/output devices respond to the addresses used on the Nascom. There is thus no danger of the programs incorrectly affecting the input/output devices.

All signals leading onto the bus are buffered by LS TTL 240 family devices. The board therefore represents a known bus load which is extremely low and has the advantage of hysteresis on all signals. The lines that are open collector are terminated with a 680 ohm pull up resistor on the edge of the card and are driven onto the card as normal totem pole output signals. The required combination is performed by logic gates introduced because of the differences in active level amongst the several devices on board.

6.6 Read Only Memory

The major use of read only memory in computer systems is: firstly, to provide non-volatile program storage where there is no convenient means of loading programs into random access memory after power has been applied, secondly, to provide an unalterable program copy for use in systems where the possibility of program corruption is high (high noise environments for example) or finally, to maintain a debug system that is available to the programmer should more usual program loading methods fail. When building any computer system it is necessary to establish various parameters before deciding on the amount of read only memory to be provided. A certain minimum is clearly needed for a hardware monitor to take control of the system on power up, and such control might consist of waiting for keyboard commands or booting a more comprehensive operating system from disk or other high speed program loading peripherals. If however, the system lacks any high speed

program loading devices, various large pieces of supportive software may have to be located in read only memory to prevent excessive delays on occasions when such software is required. For example, in the system under discussion the cassette tape interface operates at three hundred baud, approximately thirty characters per second. Assuming that the data is written to the cassette recorder in pure binary, an eight kilobyte program would take over four and a half minutes to load. This assumes that data retrieval is perfect. As this is rarely the case, data is stored and retrieved in blocks so that an incorrectly received block can be reloaded without restarting the entire load sequence. Since data is also protected by checksum, the net effect is to further extend the time to load such a program.

Having determined that it is necessary to provide a faster means of loading such programs into the processor's memory, the programs to benefit from such storage must be identified and it is necessary to decide which facilities should be instantly available.

As mentioned above, a monitor program for the manipulation of memory and registers, offering debug facilities, is a prime requirement and this is furnished by the two kilobytes of EPROM on the processor card. The other major facilities that justify the expense of storage in this manner are a high level language or the rapid generation of simple programming examples and calculations and some form of assembly language assembler and editor. Software already available for similar Z80 systems was examined and a BASIC language program was found that occupied one eight kilobyte read only memory, whilst an assembler editor requiring four kilobytes was also available. Apart from the monitor, this indicates a requirement for at least twelve kilobytes of read only memory storage. As the Basic language interpreter came in a known ROM type, provision was made for one such device on the second card. The design of the circuitry to accommodate the four kilobytes was further complicated by the

desire to use be able to use any one of the available devices (2708, 2716, 2516, 2532) of which the 2708 was the most cost effective and common at the time the design was fixed. Two problems are introduced when attempting to design a system to use any of the above devices. The first problem relates to the different pin outs employed by the different types of EPROM. The 27XX devices require three voltage rails (+/-5, +12 volts), whilst the 25XX devices are single rail. As the larger devices require more address lines, these take the place of the voltage rails no longer required. Figure 6.2 shows the partial pin out of each of these devices for the area in which they differ. If these lines are made rewireable, it is possible to equip the board for any such device. As the most common device was the 2708, the tracks to the socket were those appropriate for the 2708, but in each case two wire-wrap pins were placed in each track, so that at some future date the track could be cut and new wire links made to nearby posts holding the alternative signals could be substituted. This arrangement is shown in Figure 6.3.

The second problem with multi-use sockets relates to the generation of chip enable signals. Here, three possibilities present themselves. In the first case, the address space given to each socket is that of the largest device that can inhabit it. This would involve using sixteen kilobytes of the address map and, if the sockets contained 2708s, each device would image four times. Whilst the memory management unit enables the loss of sixteen kilobytes to be viewed with a certain amount of equanimity as the available space is one megabyte, such a scheme does introduce problems for the programmer, who must cope with the non-contiguous address spaces provided. The second alternative is to provide an address space suitable for four of the smallest devices (2708), thus giving four kilobytes, and as larger devices are used, reduce the number allowed. This would enable boards to contain four 2708s, two 2716s or one 2532. As the larger devices become more common, the number of boards that

are part populated will increase. The final option is to alter the address decode circuitry in line with the type of EPROM employed. Now, the total space allocated to the EPROMs will be four, eight or sixteen kilobytes when 2708, 2716 or 2532 type devices are used. This final, and more desirable option is the one in use for this system. Once the EPROM block is selected, the generation of one of the four chip enable signals is performed by one section of a 74LS139, the two selector lines normally being connected to A10 and A11, thus giving one kilobyte per socket and hence being suitable for 2708 devices. Again, these lines are supplied with two wire-wrap pins with a track that can be cut so that higher order address lines (up to A13 and A14) may be substituted, implying eight kilobyte devices in the sockets. As one LS139 is used, the space provided to each socket will be identical, even though it is possible to wire individual sockets for different types of EPROM. Initially the four sockets are provided with a four kilobyte block, which the LS139 splits into four one kilobyte sections. If larger devices are employed, and the extra space is required, an additional sixteen kilobytes of address space can be allocated to the EPROM bank, giving twenty kilobytes in total. This additional sixteen kilobytes is allocated to the dynamic random access memory devices by default, but, if the larger EPROM address space is required, the board must not contain the forty eight kilobytes of RAM that is possible, rather the total is reduced to thirty two kilobytes of RAM. The method of reassigning this additional sixteen kilobytes of space is detailed in Section 6.9.

6.7 Random Access Memory

The function of random access memory in a system is, like read only memory, dependent on the type of peripherals attached to the processor. If peripherals are of a type not suitable for program entry (analogue to digital converters, sense switches etcetera) or are extremely slow (audio cassette,

teletype paper tape reader), and therefore not viable as program storage devices, the majority of programs will be stored in read only memory. Such random access memory as is present in the system will be for data and variable storage only. For this reason, most single chip microcomputers have a ROM/RAM ratio of approximately 8/1 as little variable storage is required for control tasks. Certain programs manipulate large quantities of data, but such programs are editors, assemblers and language compilers, which require efficient data retrieval for useful functioning. If such peripherals exist, the amount of random access memory can be dramatically increased to allow the loading of programs. The language interpreters (of which BASIC is the most common example) represent an intermediate stage, and the program can allow the user to generate extremely compact code that can accept or generate large quantities of data. In this system there is one peripheral capable of extremely high rates of data generation: the slave processor. If the slave is single stepped at high speed, the program flow and register information can usefully be saved for review on program completion. This requires a large random access memory space, whilst the ability of the slave microprocessor to access the Z80 bus allows the direct utilization of the Z80 random access memory for storage of data collected or generated by the slave. Finally, the presence of a Basic interpreter and the ability of the system to support floppy disk and the CP/M operating system also enables a large random access memory to be fully exploited. For this reason, the random access memory space is the largest element of the memory map at forty eight kilobytes.

As has already been mentioned, the Z80 has the ability to supply the refresh cycles needed to maintain the contents of dynamic random access memory. This is achieved by the use of a counter internal to the processor which is incremented on each instruction fetch. The value produced by the counter is then placed on to the address bus immediately after the op-code

fetch of each instruction and at the same time the $\overline{RFSH}$ line becomes active. Dynamic random access memory is approximately one quarter of the cost per bit of static memory, though this advantage can be lost if refresh circuitry is required. For Z80 systems, refresh circuitry does not have to be considered and the increased complexity associated with the use of dynamic random access memory is limited to the production of a multiplexed set of address lines and the related address strobe and address multiplexer control signals. A further complexity introduced because of the nature of the Z80 control signals is caused by the late generation of the write signal ($\overline{WR}$).

Dynamic random access memory is most commonly found in high performance computer systems or those employing error detection and correction circuitry. In both these cases, the read/modify/write bus cycle offers major advantages in that the address set-up and multiplexing time is not required for the write part of the cycle. To cater for this market, the dynamic random access memory designers allow three basic types of access cycle. To control the selection of the appropriate cycle it is obviously necessary to have more than one signal. As this is not possible without an increase in the number of pins and therefore cost, the relative timing of the write signal with respect to the two address strobe signals, row address strobe ($\overline{RAS}$) and column address strobe ($\overline{CAS}$), is used to select between a read only cycle, a write cycle and a read/modify/write cycle. The normal timing of the Z80 write signal is such that the read/modify/write cycle is selected. As the dynamic random access memories have separate input and output pins, the read/modify/write cycle accepts data and presents it simultaneously. If the input and output pins are both connected to the data bus, the simplest and most desirable method, a conflict will occur on the data bus between the processor generated data and that generated by the dynamic random access memory. Therefore, either the dynamic random access memory output must be independently buffered and enabled

by the read signal, or the timing of the write signal must be altered to cause selection of the write only cycle. In fact, the generation of the correct control signal is preferable as this eases the generation of the control signals associated with the data bus buffers. Examination of dynamic memory timing diagrams[15] shows that if the Z80 write signal were to operate with the timing of the read signal, the dynamic random access memories would function satisfactorily. This can be achieved by using the presence or absence of the read signal ($\overline{RD}$) in conjunction with the $\overline{MREQ}$ and $\overline{IORQ}$ signals. The resultant signal is known as $\overline{MWR}$ (modified write).

The other cycle type of the dynamic random access memories that is of importance in this application is the refresh cycle. A refresh cycle is any in which the $\overline{RAS}$ signal is activated and then removed without the simultaneous activation of the $\overline{CAS}$ signal. To ensure the maximum refresh rate, the $\overline{RAS}$ signal is tied to the $\overline{MREQ}$ signal as generated by the processor. This ensures that any memory bus access performs a refresh cycle, the penalty being an increase in the power consumed by the dynamic devices. The generation of the $\overline{CAS}$ signal is therefore the function of the chip selection circuitry and is detailed in Section 6.9.

6.8 The Memory Management Unit

The memory management unit represents the most complex section of the two boards that comprise the basic system and in the entire system is equalled in complexity only by the logic of the interface board. The major elements having been described in Chapter 5, this section will deal with the logic responsible for several specific functions.

As mentioned, the bipolar random access memories used for address translation were selected for high speed and therefore are more sensitive to any noise present in the system. This is manifested by their ability to complete a full write cycle whilst the write control signal is moving from the

active to the inactive state. As the bus buffers are disabled by a signal that is used to generate the write signal, the schottky buffers are disabled fractionally before the removal of the write signal. Due to bus capacitance and the relatively low speed of NMOS devices, this does not cause difficulty throughout the rest of the board. A special write signal is therefore generated for the bipolar devices by the use of two cascaded D type flip flops, the circuit being shown in Figure 6.4. The two input signals $\overline{MWR}$ and $\overline{ONCARDIO}$ are the modified write and bipolar random access memory select signals respectively and the resultant waveforms are shown in Figure 6.5. As can be seen, the resultant write signal applied to the bipolar memories is a single, short duration pulse in the middle of the bus cycle.

Part of the memory management function is the generation of wait states as demanded by particular blocks. This is achieved by a similarly cascaded set of D type flip flops where the input is a combination of the $\overline{MREQ}$ and $\overline{M1}$ signals ORed with their respective wait state request signals as generated by the attribute memories. Thus if a wait state is requested, the $\overline{WAIT}$ line will exhibit the same timing as the bipolar random access memory write line. As the $\overline{WAIT}$ line is sampled on the falling edge of the system clock, the circuit guarantees the insertion of one wait state on any cycle of the appropriate type.

The memory manager must, when not enabled, generate only Z80 addresses and to achieve this the additional address lines must be forced to zero. This is accomplished by enabling and disabling the buffer which supplies these lines (A16 to A19) to the board and the rest of the system. As the signals are in inverted form on this card, the outputs of the buffer are tied to the positive rail with one kilohm resistors, and, when the buffer is disabled, the inverted address lines are taken high thus producing the desired result. The enable signal for the buffer is a simple combination of the two signals that

disable the production of a high order address, namely the disable memory management unit ($\overline{INIT}$) and bus acknowledge ($\overline{BACK'}$) signals. The additional address lines must be disabled under direct memory access conditions (i.e. slave accesses) as there is no translation memory to supply these addresses. By implication, the slave can only access addresses from 0 to $0FFFF_{16}$, these addresses being completely occupied by devices on the first and second cards.

As stated in Chapter 5, the selection of the correct address by the memory manager (untranslated, reset offset vector, translated, fixed value for input/output cycles) is performed by four to one multiplexers, 74LS253. The generation of the control signals for these multiplexers is in turn performed by a priority encoder device, the 74LS148. This offers several advantages in that the encoding necessary for the multiplexer is performed and, when two conflicting requests are fed to the encoder, the problem is automatically resolved in a consistent and predetermined manner. The circuit is shown in Figure 6.6 and it can be seen that the LS148 provides eight input lines (0 to 7) where a higher number will override all lower numbers. Three binary output lines are provided which encode the number of the highest active input. Of these, the two low order bits are used, therefore there will be, for each possible address source available to the multiplexers, two lines which, when active as the highest priority, will select that source. The priorities and the signals they are allocated to are shown in Figure 6.7. As can be seen, the reset vector holds the highest priority and will therefore, for the duration of the first instruction, force the offset onto the high order address lines. The $\overline{IORQ}$ signal is used to select the second highest priority. The memory management unit disable signal has a priority of one, whilst the normal address translation active state has the lowest priority. Priority five, which selects the same address source as the memory manager disable signal is

provided with a pull up resistor and switch. If this line is taken low, the action of the memory management unit will be inhibited (except for reset and input/output cycles) even though software has enabled the memory manager. This facility is provided for debugging purposes as software can be allowed to execute as normal, but any operations involving the memory management unit will not affect the memory map. This is of particular use when the software being debugged is responsible for manipulating the memory manager as debug software can then be used to check the contents of the translation and attribute memories, even though the contents of the memories would normally remove the debug software from the memory map.

6.9 Board Two: Miscellaneous Functions

The function of the second board as a gateway to the slave and expansion box address spaces introduces extreme complexity into the circuits responsible for buffer control. On a normal card, the address buffers will point onto the card only and are always enabled. The data buffers will be enabled when any device on the card is selected, usually by means of a board select signal. The direction of the data buffers is then controlled by the read or write signals or a simple combination of the two. With the second board there are many more possibilities for both the source of the address and the location of the responding memory device. These transfers can be summarized as follows: -

Address Source	Data Source/Destination
Z80	Z80 system
Z80	Board 2
Z80	Slave/Expansion box
Slave/Expansion	Slave/Expansion box
Slave/Expansion	Board 2
Slave/Expansion	Z80 System

Further difficulties arise when the interrupt system of the Z80 is

considered, in that all devices that can use the Z80 vectored interrupt scheme must listen to the data bus so that the execution of a return from interrupt (RETI) instruction can be detected, which is used to control the daisy chain between the peripherals. Therefore, the board must transmit the contents of the Z80 bus whenever possible so that any peripherals on the interface or in the expansion box can monitor the instruction stream. When an interrupt is acknowledged, the interrupting device generates a vector that must be transmitted to the Z80, and the board must have some mechanism that comes into operation during interrupt acknowledge cycles which will correctly place the location of the interrupting device and control the data buffers accordingly.

The generation of the buffer control signals is achieved by the production of a limited number of signals which are then fed to a bipolar programmable read only memory as address lines, the data outputs being used to both enable and direct the buffers on both the Z80 and interface/expansion box connectors of the card. The signals fed into the read only memory are the read, write and bus acknowledge ($\overline{RD}$, $\overline{WR}$, $\overline{BACK'}$) signals, a derived signal indicating that the addressed device is on the second card and a signal that indicates that the selected device is beyond board 2, i.e. slave, expansion box or interface (shared memory). The last signal mentioned would normally be impossible to generate, but as these locations can only be accessed by setting the appropriate attribute bits in the memory management unit, examination of these bits provides information as to the location of the currently addressed memory device. To cope with the necessity for identifying the location of an interrupting peripheral, the interrupt daisy chain itself is examined. Devices in the expansion box and on the slave interface are defined to be at the top of the daisy chain, thus if the chain is activated to prevent interrupts by devices lower in priority, the responsible device must be beyond the second board.

The last two signals fed to the programmable read only memory identify the addressed location as either being on the second card or beyond it and, if neither of these signals is active during a memory access cycle, the addressed location must by a process of elimination, be located on the Z80 system bus, and the data buffers of the second card need take no action whatsoever. If the slave is performing the access, indicated by $\overline{BACK'}$ being active, the only two possible locations are board 2 or board 1, which will again be indicated by the presence or absence of the "on board 2" signal.

To allow flexibility in the positioning of input/output devices, two bits are reserved in input/output location 44_{16} to indicate the position of input/output addresses 80_{14}-BF_{16} and $C0_{16}$-FF_{16}. The user indicates that these areas do (or do not) require board 2 activity when accessed by the processor. No attempt is made by these signals to extend the input/output space.

The use of the programmable read only memory as a source of data buffer control signals is convenient as the output state can be completely defined for each possible combination of the five input signals. This relationship is shown in Table 2. The ability to generate new relationships in the future to meet any change in design without the need to alter the component count or package is the major advantage of all the programmable logic elements. The control of the address buffers is much simpler, the direction being given by the state of the $\overline{BACK'}$ signal (active, towards the Z80), which is also used to control the state of the buffers responsible for the control signals that must be driven by any device performing direct memory access: - $\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$.

The generation of the selection signals $\overline{NATIVE}$, $\overline{EXPBS}$, $\overline{SMR}$, $\overline{OMS}$, $\overline{OMIOSEL}$ (or $\overline{TIOS}$) is also performed on the second card. These signals are derived from the attribute RAM of the memory management unit and represent memory contained within the Z80 system, memory in the expansion box, memory on the interface card (shared memory), memory within the slave microcomputer and input/output

devices in the slave system (target I/O space), which are treated by the Z80 as memory devices. As all interface cards require at least one parallel input/output device, these signals are further modified so that during input/output cycles they provide most of the decode necessary. The major reason for implementing the interface decode logic on the second card is that it is theoretically possible to have memory management extension circuits on both a slave interface and expansion box interface. Both these circuits should share a common address for up-date purposes so that the bulk of memory manager software can operate in ignorance of the two sets of circuitry. Where it is necessary to uniquely select one interface, this can be performed by the top bit of input/output location 44_{16}. Software is thus able to function in the same fashion irrespective of the system configuration. Therefore, if $\overline{\text{SMR}}$ or $\overline{\text{EXPBS}}$ become active *without* $\overline{\text{MREQ}}$ but with $\overline{\text{IORQ}}$, the interface control devices on the relevant card are selected. All the selection signals are further modified in that if an address is the subject of a write cycle, and that address is write protected, the selection signal will not become active. As the address decode for all devices must include one of these signals, no device will respond to the write cycle. The $\overline{\text{MREQ}}$ signal of the Z80 is also subject to modification in that if the $\overline{\text{OMIOSEL}}$ signal is active, the $\overline{\text{MREQ}}$ signal to the interface is removed as the access demanded of the interface is not of memory as implied by $\overline{\text{MREQ}}$, but of an input/output device. For microprocessors with separate memory and input/output address spaces, the two signals $\overline{\text{OMS}}$ and $\overline{\text{OMIOSEL}}$ represent analogues of $\overline{\text{MREQ}}$ and $\overline{\text{IORQ}}$.

The decode circuitry for on board memory and input/output devices each utilize a programmable read only memory of the same type as that described for the data bus buffer control (74S288 32 byte PROM). This approach was adopted as it provides the flexibility to allow several different configurations to be provided with wire link selection, without excessive package count. In the

case of the memory decode, each output of the programmable read only memory is followed by a NAND gate as the S288 generates glitches on the data outputs that cause spurious address latching on the dynamic random access memories, the NAND gates apply a board select signal and also filter out the unwanted pulses. The programmable read only memory takes as address lines A12 to A15 and a delayed version of the $\overline{\text{MREQ}}$ signal, although $\overline{\text{MREQ}}$ is included with the board select signal that is applied after the PROM. The delayed $\overline{\text{MREQ}}$ is generated by an inverter chain attached to the normal Z80 $\overline{\text{MREQ}}$ signal. As described in Section 6.7, the dynamic random access memories require two address latch signals. The first of these ($\overline{\text{RAS}}$) is provided by the $\overline{\text{MREQ}}$ signal itself whilst a signal delayed by three inverter propogation delay times is then used to control the address multiplexer attached to the dynamic random access memory, more delay occuring before the signal is presented to the programmable read only memory. If the address lines indicate that a dynamic random access memory is to be selected, the first part of the address will be latched in by $\overline{\text{MREQ}}$, the address multiplexer will change and give the lines additional time in which to settle to the new value and the PROM output will be taken high. If the other NAND input (a combination of A16 to A19, $\overline{\text{NATIVE}}$ and $\overline{\text{RFSH}}$) is also high, the appropriate $\overline{\text{CAS}}$ signal will become active, completing the selection process. This pattern is repeated on three of the output pins of the programmable read only memory.

One pin is used to generate the chip selection signal of the BASIC language read only memory, again in combination with the board select signal and also the write signal. This ensures that the read only memory will not enable its output drivers if by mistake it is written to. This circuitry is also used for the EPROM selection signal, two pins being provided, one generating a four kilobyte block, the other twenty kilobytes. If the later connection is made, only thirty two kilobytes (two banks) of dynamic random

access memory can be placed on the board. The last signal generated by the read only memory is used to indicate that the memory device selected is on the second board. Again two options are provided, one assumes the full forty eight kilobytes of dynamic random access memory or thirty two kilobytes of dynamic memory and twenty kilobytes of EPROM, whilst the other does not generate a signal for the last sixteen kilobytes of dynamic memory, thus allowing this address space to be used by other boards in the event that board two is only partially populated.

The input/output decode programmable read only memory is used to generate the chip selects for all the bipolar translation memories (address and attributes) and their appropriate data buffers. The programmable read only memory receives the signals A0, A1, A2, an on-card input/output signal and the $\overline{\text{BACK'}}$ signal. As outputs it generates the selection of the correct bipolar random access memories when both normal and direct memory access memory cycles occur (i.e. when $\overline{\text{ONBOARDIO}}$ is not active) and during memory manager update and access operations take place, the correct bipolar memory and buffer are opened to allow the read/write cycle to take place. The programmable read only memory is convenient as a decode circuit in this application as the translation memories are activated in pairs during memory accesses (master or slave translation and attribute), yet when accessed as input/output devices, they must be enabled separately. The control signals for these devices are therefore generated by a read only memory. The action of this read only memory under various conditions is shown in Table 3.

6.10 The "Softy" Board

When a microcomputer system is being examined, there are several objects of attention. Of these, the most crucial to the proper understanding of the microprocessor are, the current values of the registers, the area of the stack, the section of the memory in the immediate vicinity of the current

instruction and any memory locations addressed by the current instruction. Whilst much attention has been devoted to the method used to display the registers, few systems offer facilities to automatically display any location being referenced by the instruction stream. An alternative approach, used within this system, is to offer a real time display of a limited memory area, and allow the user to concentrate the processor stack, and any variables of interest, within this area.

This approach offers several distinct advantages over either display on demand or dynamic display of processor referenced locations. The display mechanism is automatic and this is essential if the user is not expecting any changes in the memory area. If the area is constantly displayed, the ability of the human eye to detect motion will reveal a change within the display area, even if unexpected. This can be further enhanced if any up-dated locations are highlighted for a predetermined period. The advantages offered over dynamic displays are of constancy and depth. Whilst the ability to preview the contents of any location that will be referenced by the current instruction is of major benefit, the facility is not extended to other information which is logically connected with the current instruction. For example, if two ten digit numbers are being added, a dynamic display will offer the register contents and the location about to be added into the register, but not those locations that have been or will be added. If the display is widened to show several locations on either side of the current focus of interest, the problem of correctly identifying the function of a particular location is increased as the location will move relative to the current focus and therefore will also move relative to the reference frame of the display device. The net effect is that as time passes, the user will have to look at different sections of the screen to locate the same object. These problems all relate to displays generated during single stepping. There is no

ability to generate a display based on the contents of memory as a processor runs at full speed unless the display function is directly supported by hardware rather than software. Whilst the usefulness of such a display is impaired if the processor is up-dating locations at high speed, it is possible to identify those locations which are currently the focus of processor activity.

6.10.1 Facilities

The Softy card is based upon a commercial design of the same name. The function of the commercial version is the preparation and display of data to be programmed into, or taken from, 2708 erasable, programmable read only memories. To achieve this a one kilobyte random access memory is provided which can be manipulated by an on board INS8060 microprocessor. The processor can accept data from a keyboard, load from or store to cassette tape, program the EPROM with the data held in the memory buffer or copy the contents of an EPROM into the memory buffer. The display presented to the user is a screen composed of thirty two rows, each comprising sixteen hexadecimal character pairs. Thus, at any instant, the screen displays the contents of 512 bytes. Examination of the other half of the RAM buffer, or either half of the EPROM currently in the programming socket is accomplished by attempting to move the screen cursor above or below the current screen image. Such action places the cursor at the bottom (or top) of the next page in sequence. Extensive circuit modifications have been made so that the board has two kilobytes of memory buffer which can be organized as two kilobytes or one kiloword. The design of the system is such that these buffers can be displayed as byte pairs, or the screen can display either the upper or lower half of the sixteen bit memory, prior to placing this data into an EPROM. Furthermore, the board can be accessed by a wide range of microprocessors and is accommodated to individual bus timings by a "personality module" (specific to a particular processor)

that is located in a twenty eight pin socket on the board. When placed within a system, the microprocessor can access the Softy as normal random access memory and, when not being accessed, the display circuitry transmits the contents of each byte as two hexadecimal digits on screen. With the proviso that the microprocessor has absolute access priority and any attempts to force the on-board INS8060 to execute programs is likely to fail if the external microprocessor accesses are too frequent, the INS8060 can be used to alter the displayed page or up-date memory locations by the use of its own keypad.

The Softy board can be divided into four main functional areas, these being the processor, the buffer used to assemble the data to be placed in EPROM, the display circuit and the EPROM programming circuit. All the modifications mentioned are related to the circuitry of the random access memory buffer and the effect of the added circuitry is to produce a buffer which operates in an identical fashion in any interaction with the rest of the board, yet offers superior facilities to external processors. The commercial form of the circuit will therefore be described in detail followed by a detailed description of the new form of the random access memory buffer.

6.10.2 The Softy Processor

The processor chosen for the Softy was the INS8060. Whilst this processor has a limited instruction set and is not able to offer high speed instruction execution, it has several facilities which are extremely valuable in this application. Likewise, the stated limitations are not significant in the context of this application. The INS8060 has a sixteen bit program counter, but the address bus is only twelve lines wide. The high order four bits of the address bus are placed onto the data bus at the start of any bus cycle. Because of this, although the INS8060 can address sixty four kilobytes, this design limits the address map to the four kilobytes available without the use of address latches.

The address map as seen by the processor consists of four, one kilobyte, sections. The first, at address 0 contains the firmware of the system in this position as the INS8060 starts execution at location 1 after a processor reset. The board has an INS8154 decoded as the next kilobyte and this leads to several images as the INS8154 only offers 128 bytes of random access memory and two input/output ports. The random access memory is used for the storage of temporary variables associated with processor execution, whilst the input/output ports are used to provide a keyboard interface and to control the screen display. The EPROM to be read or programmed is decoded next, between addresses 0800_{16} and $0BFF_{16}$. Finally, the random access memory buffer decodes to addresses $0C00_{16}$ to $0FFF_{16}$.

A major advantage of the INS8060 in this application is, as stated in Chapter 4, it is not designed to be the controlling device in the system. As the video circuit employs the same address and data bus as the INS8060, the processor cannot use the buses during screen display. To achieve this, the processor is halted except when executing a command. This design uses the CONT line of the INS8060 which, when taken low, suspends all processor activity. The CONT line is driven by an eight input NAND gate, connected to the outputs of the keyboard scanning matrix and a one bit output port (F2) of the processor. When a command is completed, the processor outputs a low onto all keyboard scanning inputs, takes the F2 line high and, if no keys are pressed, the processor will halt. This situation will continue until a key is pressed, at which time the NAND output becomes high and execution recommences. The first action of the processor is to set F2 to zero volts so that execution may continue after the key has been released. The video circuit is enabled by the NENOUT line of the processor, so that when the processor halts, control of the buses are passed to the video generator and a coherent display can be placed on the screen.

The serial input and output available from the processor is used to produce the cassette interface. The output is taken, via a resistor, to the microphone input of a cassette recorder, whilst the output of the recorder is fed into the input of a CMOS exclusive OR gate which is biassed into the linear region. Due to the high input impedance of the CMOS input, the input voltage swing is large and the gate acts as a waveform shaper. The output from the exclusive OR gate is fed into both the serial input of the INS8060 and also a single bit input port. The cassette format is totally asynchronous, using variations in the time between signal transitions as the as the indication of a one or a zero. The resultant ability to store one kilobyte of data in approximately five seconds gives an effective baud rate of ten kilobaud, extremely high for cassette storage.

As mentioned, the keyboard matrix of the system is organized as three rows by seven columns. The seven inputs are connected to one of the ports provided by the INS8154, whilst the three outputs are driven by the second INS8154 port. To locate the currently pressed key, one output line is taken low and the input line corresponding to that column will go low only if the key on both the row and the column is pressed. This is repeated for the other two keyboard rows. Of the possible twenty one matrix positions, twenty are occupied by scanned switches and the last is occupied by the processor reset key.

The functions provided by the software of the processor are largely to perform data manipulation. By a single command, the user can copy the contents of either the firmware EPROM, or the EPROM to be programmed, into the random access memory buffer. As EPROMs contain FF_{16} when empty, a command is available that fills the buffer with this value. In this way, only sections of the EPROM that are to be used will be programmed, other sections may then be programmed at a later date without first erasing the EPROM. Data is entered, a

byte at a time, into the buffer by pressing two hexadecimal number keys in sequence. The location of the current position being written to is shown by a highlighted cursor on screen. The position of this cursor may be moved backwards or forwards by the appropriate cursor control key, and if the cursor control key is held down for the length of time taken to move sixteen locations, the cursor will move up or down a column, rather than across a row. Commands also exist to define a block of locations on screen so that the block may then be moved backwards or forwards, or copied to the current cursor position. The entire RAM buffer can be copied to or loaded from tape and the EPROM programming function is invoked by a single key depression.

6.10.3 The Random Access Memory Buffer

In the commercial version of the Softy, the RAM buffer is relatively straightforward. When selected by the processor, the memory may be read or written. The buffer is nine bits wide, eight bits being used for data storage, whilst the ninth is used in the generation of the cursor. Whenever data is written to the random access memory buffer, the value of a processor single bit output (F0) is written into an MM2102, 1024 by 1 bit memory. Whenever the video system displays the screen, this value is read out and highlights the currently displayed byte. In this way, the entire screen can be highlighted if necessary, as can any section of the screen. It is in this manner that blocks of memory to be moved, badly programmed locations or those locations found to match the search byte in the comparison instruction, are indicated. To set the position to highlight, F0 is taken high, the contents of the random access memory at the desired location are read then written back, and F0 is then taken low.

As designed, the random access memory buffer could be accessed by an external processor so that programs could be developed in random access memory and tested before being committed to erasable programmable read only memory

since programming these devices is a time consuming process. To achieve this, the external processor must be able to access the Softy and this is done by wiring the address and data buses of the INS8060 onto a header plug that has the same pinout configuration as a 2708 EPROM. This is a form of in-circuit emulation, but here the memory device is being emulated rather than the microprocessor. Such devices have come to be known by a contraction of 'Read Only Memory emULATORS' or romulators.

To gain control of the address and data buses, the external device must be able to force off the bus both the INS8060 and the video circuit. This can be accomplished by the use of the NENIN line of the processor, which disables both the processor and any device attached to the NENOUT line (in this case the video circuit). A problem with the commercial design is that no buffers exist to isolate the data and address buses of the INS8060 from those of the system to which it is connected. Therefore the INS8060 and video system must always be disabled when connected into a system, which prohibits the generation of a meaningful display, even when the Softy 'ROM' is not being accessed.

As the INS8060 only has an eight bit bus, if programs are to be developed for a microcomputer with a sixteen bit wide data bus, two Softys are required. The alternative approach is that program development be limited to either the top or bottom half of words whilst the other half of the word contains a fixed EPROM, obviously an unacceptable arrangement.

6.10.4 Softy Display

The display of the Softy consists, at any one time, of thirty four lines, each consisting of thirty two hexadecimal characters arranged in pairs. The top and bottom of the display represent data held in the INS8154 RAM I/O chip and are used to show various parameters. The remaining thirty four lines show the contents of 512 bytes of the buffer RAM or EPROM to be programmed. The

lines containing information from the INS8154 are shown as white characters on black whilst those from the memory buffer show black on white or grey. The 512 bytes is divided into four sections by variations of background brightness, the division into 128 byte blocks allowing the limit of relative jumps to be quickly located and generally assisting in the location of a given address. Any location may be highlighted in combination with any other locations, by the mechanism of the ninth bit of the buffer as described in the previous section. To display a location not currently on screen, the cursor is moved off screen in the desired direction, whereupon the presented data will change and the cursor re-appear at the opposite end of the screen, at the next location in sequence.

The necessary waveforms for the production of a video display are derived from a video timing chain. Starting with the 4MHz microprocessor clock, which is fed as the operating frequency of the shift register, an 800KHz signal is derived. This is used to derive the select input of a 74LS157 quad two to one multiplexer that is connected to the data bus. The multiplexer selects the alternate nibbles of the data bus for display. After further division, the four address lines used to select memory locations to be displayed on one row are derived, thus giving 16 bytes/line. These values are generated as part of the cycle of a divide by twenty five counter, composed of a 74LS93, a 74LS73 and several logic gates. The result is that one sync pulse is generated every 62.5 microseconds. The correct rate for a line sync pulse is 64 microseconds and the circuit relies upon the good will of video monitors to accept this signal. The line sync pulses are fed into a CD4040 twelve stage counter, which is basically counting television lines. The bottom three stages of the counter are used as part of the address fed into a 256 by four bit bipolar RAM (74S287) which is used as a character generator. Four more address lines are used to select the particular hexadecimal characters to be displayed by the

programmable read only memory, and these lines are taken from the output of the 74LS157 mentioned above. The use of only seven lines indicates that the programmable read only memory is only half used. The three programmable read only memory address lines produced by the CD4040 indicate which line of the character is to be displayed. The output of the programmable read only memory is fed into the video shift register (7495) whose load pulse is derived from the high/low nibble control waveform of the 74LS157. The CD4040 outputs are also used, with those generated earlier in the chain, to produce a memory address. The counter produces A0 to A8 giving the five hundred and twelve locations seen on the screen. Also A11 is generated which is used to display at the top and bottom of the screen, the contents of the INS8154 random access memory, particularly those sections that hold usable information, such as cursor position. Lines A9 and A10 are provided by the INS8154 I/O port. It is these lines that the processor changes the value of to effect a change in displayed page.

All the addresses generated for video display purposes are buffered by two CMOS three state drivers (CD4503). These are controlled by the NENOUT signal as generated by the INS8060. The frame sync is provided by logic gates connected to the output of the CD4040. The sync, video shift register output, cursor and background signals are combined by the circuit of Figure 6.8. The diode D1 is used to pull the summing point to 0.7 volts which represents black level, and is hence a flyback blanking control. TR2 is used to pull the summing point to zero volts for sync pulses and the character information is fed through R10 from an XOR gate used to generate black on white or white on black dependent on whether buffer or INS8154 data is being displayed. The resultant waveforms are then fed into a UHF modulator which provides the output from the system.

6.10.5 EPROM Programming

The 2708 EPROM can be programmed by presenting stable address and data information onto the relevant pins, placing the $\overline{CE}$ signal at twelve volts and applying twenty seven volts to the Vpp pin. This information must be held for a millisecond and then the process repeated for the next location. Once all locations have been accessed, the process must be repeated one hundred times. The earlier (1702) and later (2716, 2532) devices could be programmed a location at a time. If this is done with the 2708, the device will be destroyed. The processor must therefore have some means of latching the address and data buses and of timing a one millisecond period. As the INS8060 is a static device and has the facility for accessing slow memory devices, the use of a 555 timer to generate a 'wait for the slow memory' signal (NHOLD) will force the data and address buses to remain stable for the required time. A 555 configured as a retriggerable monostable is used to generate a one millisecond pulse which allows twenty seven volts to be fed into the EPROM providing that the EPROM is being written to. Similarly the 555 output is also conditioned by the EPROM write select so that normal memory accesses will not require one millisecond to complete.

To program an EPROM, the processor copies the data held in the random access memory buffer one hundred times. At each write access, the 555 is triggered and holds the processor address and data bus station for the one millisecond. To prevent accidental writing of the EPROM, the triggering mechanism requires that the single bit output F1 be set to 1 before any action takes place.

The board can also be used to program the three rail variant of the 2716. As the address space available for EPROM programming cannot be increased without exceeding the four kilobyte limit, a 2716 must be programmed as two, one kilobyte devices. The half of the 2716 to be affected being selected by a

switch on the board. A further change required is the substitution of one programming pulse of fifty milliisecond for one programming cycle rather than one millisecond for one hundred cycles.

In all cases, once the EPROM is programmed, it is compared with the contents of the random access memory buffer and any differences highlighted. Such differences will occur if the EPROM was not empty or is damaged in some way.

6.10.6 Softy Modifications

The modifications undertaken to the design of this board can be broadly split into two halves. The first set of modifications are designed to enable a section of memory within an INS8060 system (the random access memory buffer) to be accessed by an external processor that may have either an eight or sixteen bit data bus so that new data may be placed in the memory, or old data read. The board must be designed so that as diverse a range of control signals as possible can be accommodated with the minimum of expense. Buffering must be provided so that the information held in the random access memory buffer is available for inspection through the video system whenever possible. The second set of modifications present are those that enable the correct presentation of data to the INS8060 and the screen circuitry. It is desirable that sixteen bit data should be presented as alternate byte pairs on screen and should be alterable as such but, for the purposes of programming EPROMS, it should be possible to separate all the high byte information from the low byte information and present either. Both sets of modifications were implemented so that as little as possible of the original, proven design needed to be altered. With two simple exceptions, all the modifications surround the random access memory buffer circuits. The exceptions will be discussed first before the general discussion of the random access memory buffer modifications.

When an external processor accesses the random access memory buffer, it must gain control of the Softy address, data and control buses. This is done by taking the processor NENIN signal high. This in turn takes the NENOUT signal high which, should they be enabled, disables the video timing chain to address bus buffers. As the buffers are CMOS devices and the INS8060 is NMOS, the resultant delay between the external processor requesting the bus and the video timing chain being removed, caused address bus conflict that led to data being occasionally misread. To counter this, the NENIN to NENOUT delay was reduced by placing an OR gate around the INS8060 as shown in Figure 6.9.

The second problem associated with external accesses again relates to the speed of on board devices. In the course of its operation, the video timing chain accesses the INS8154 random access memory I/O peripheral for reading data out to the screen. The accesses to the input/output port of this design are incidental to the correct operation of the display, but as only a read cycle is involved, no problems arise. If the external processor acquires the data, address and control buses for a write cycle whilst the INS8154 input/output device is being accessed by the VTC, it is possible that the timing constraints of the device will not be met and an undesired write to the I/O port can take place. The observable effect of such a write is an unrequested change in the page being displayed as the value on the output port is changed. It is also possible that the keyboard scanning input values could be changed thus disabling the processor "wake up" mechanism that requires all these lines to be low for correct operation. This would require a processor reset to regain control of the system. To remove this problem, the address line from the video timing chain that forces selection of the input/output device can now be switched so that either normal or non-display of INS8154 locations can be selected. If the input/output device is never active, the difficulty does not occur as there is no possibility of an incorrect write

cycle taking place.

The ability to interface this board to a wide range of different microprocessors could be achieved in two ways. The method commonly used is the standard bus. There are many such standards (e.g. S100/IEEE696, Multibus, Versabus) and such designs are commercially sensible. Such a scheme is not desirable for an educational system as the complexity of the processor card will rise as it must contain all the necessary control signal conversion logic. Input/output devices of the processor will not be so easy to interface, as the expected control signals will have been transmuted in order that they conform with the standards, and finally, the various buses will not be available so that a comparison of advantages and disadvantages cannot be made. The second method of providing a board common to several processors is to fix the position of the power supply, address and data buses, and also to fix an area in which all control signals will be placed. The timing and type of the control signals will however, reflect those generated by the microprocessor concerned. If such a scheme is used, provision must be made for those systems that employ a multiplexed bus structure. The technique employed is the provision of a de-multiplexer card that can be used with any boards that are not designed for multiplexed operation. The advantages of this system are that complexity is limited and restricted to those cards which provide a service rather than those to be studied, such as the microprocessor card itself. The attendant disadvantage of this scheme is that "service cards" must contain an element of re-configurable circuitry that can be altered to enable the conversion of the control signals associated with the particular processor in use. This is achieved by the provision of a "personality socket" on board which takes all the signals in the control area of the bus, power supplies and such signals from the card as are required, and generates the necessary signals to be fed onto the card.

To be used with both eight and sixteen bit processors, the Softy circuitry needs to be extended in several ways. Primarily, provision must be made for sixteen bit microprocessors to write sixteen bits of data into the buffer in one bus cycle. Similarly, a method must be provided of displaying sixteen bit data on screen. The following top line of a screen will be used as an example throughout this section.

00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

Under normal circumstances, i.e. eight bit systems, location 000 would contain 00_{16} location 001 would contain 11_{16} etcetera. The data would be stored in successive random access memory locations, with the processor A0 line connected to the random access memory A0 line and so on. But, what if the above data was written by a sixteen bit processor? Now 0011_{16} would be written in one cycle and must therefore be presented to a sixteen bit wide memory. Now, when the addresses are even, (locations 0, 2, 4, 6 . . .) one random access memory is involved, when odd, the other. The INS8060 A0 line, which originally selected between one of two locations in one random access memory, is now being used to select between the two random access memory chips. Such a solution can be used with both eight and sixteen bit processors without difficulty. However, when an EPROM is to be programmed that will be used in a sixteen bit machine, the data derived from high byte locations must be sent to one EPROM, the data from low byte to another. To achieve this, the above display will have to be altered so that either

00 22 44 66 88 AA CC EE or

11 33 55 77 99 BB DD FF

is seen, dependent on which EPROM is in use.

Now, as every second location is being taken, an address transformation must occur. This problem only arises for sixteen bit processors. The net effect of this is that a sixteen bit wide random access memory is to be

provided in which addresses must occur alternately A0, B0, A1, B1 ... or from one random access memory only A0, A1, A2 ... or B0, B1, B2 .... This can be achieved by of the circuit in Figure 6.10 For eight bit processors, A(n) of the processor is connected to A(n) of the random access memory chip whilst, for sixteen bit processors, A(n) of the processor is connected to A(n-1) of the memory. This leaves A0 of the processor unconnected and it must be used in the generation of the chip enable of the two sets of random access memory. As the meaning of A0 cannot be determined in advance, it is supplied to the personality socket and the chip enable signals are fed from the personality socket.

The signals required by the Softy board for correct operation are:

i) a board select signal. This must be derived as a combination of the external address bus and address validation signals. Once generated it can be used to enable the address and data buffers and disable the INS8060 and video timing chain buffers.

ii) a read/write signal. As many different methods can be used to indicate the direction of data transfer it is necessary that they all be converted to a common form for use on board. As the random access memory devices and data bus buffers utilize a R/$\overline{W}$ signal, this is the form that must be provided by any external devices.

iii) Chip enables for the two sets of random access memory. When a sixteen bit processor is controlling the external bus, there are a variety of accesses it can perform on a memory device. A byte or word access may be performed and, as there are several techniques for distinguishing between the two types of access, this must be done by processor specific circuitry. The signals used to enable the random access memory can then also be used to control the relevant data bus buffers.

The personality socket provided on the Softy board has twenty eight pins.

There is, therefore, not enough space to pass all the high order address lines bits through the personality socket and, as the lines will be used to generate a board select signal, they must be passed through an address comparison stage. As the circuitry for this is standard, it is included on the board itself and is shown in Figure 6.11. This circuit needs an indication that the address on the bus is valid and this is the $\overline{VA}$ signal as provided by the personality socket. In turn, the board select signal is fed into the personality socket to be used in the generation of $\overline{CE}_A$ and $\overline{CE}_B$.

As mentioned previously, it is not possible to increase the address space of the INS8060 without considerable modification and only one kilobyte of the enclosed two kilobyte can be accessed by the INS8060 processor. The selection of which section is available is performed by two on board switches that control the display format (by affecting the address multiplexers) and the upper address line of the random access memory buffer. These are called the Format and HI/LO signals respectively. When the eight bit format is selected, the address line from both the external and INS8060 processors are passed to the random access memory. The 74LS158 multiplexer that operates on the chip enable line feeds the value of the HI/LO switch and its inverse respectively to the two random access memory banks. The outputs of the 74LS158 are then only enabled when the address decode for the random access memory is generated by either the external or INS8060 processors. In this mode, both processors access either one random access memory or the other, dependent on the state of the HI/LO switch. Thus the two kilobyte buffer is only accessible one kilobyte at a time. With the 16/8 bit switch in the sixteen bit format, the INS8060 A0 is used to select the chip enables of the two random access memories with the personality socket decoding the external processor high/low byte access signals and also generating the two chip enable signals. The selection between INS8060 A0 and the external processor chip enables taking

place in the personality socket. Now the HI/LO signal feeds A9 on both sets of buffer random access memory. Again, only one kilobyte is usable at any instant.

To generate two one kilobyte EPROMs for use in a sixteen bit system, the board is placed into sixteen bit mode. One kilobyte of data is sent to the Softy with the HI/LO switch in the low position. This action half fills each one kilobyte random access memory with valid data. The HI/LO switch is moved to the high position and the external processor repeats its action. Now all further accesses from the external processor are disabled by the board access disable switch shown in Figure 6.11 (part of the external processor address generator) and the board is put into eight bit mode. The HI/LO switch is then used to select the EPROM to be burnt first (high byte or low byte) and the burn is initiated. Once programming is complete, the HI/LO switch is reversed and the cycle repeated. The data bus buffer control is derived from the $\overline{\text{EXTACC}}$ and $\overline{\text{CE}}$ signals. Referring to Figure 6.12, any external access enables buffer 'A', the direction being determined by the R/$\overline{\text{W}}$ signal. Buffer 'B' will only be enabled by a sixteen bit external access (determined by the Format switch) whilst buffer 'C' is enabled in sixteen bit mode if no external access is occurring and the INS8060 accesses random access memory bank 'B' or when in eight bit format, either the external or INS8060 processors try to access random access memory bank 'B'.

The personality socket has been used to provide a Softy to Slave interface for several processors and the interface function can usually be achieved by the use of two or three small scale integration devices.

The pins allocated to the personality socket are shown in Figure 6.13, whilst Figure 6.14 shows the interface required for an M6800 slave. The valid address signal is generated by the NANDing of VMA and $\phi 2$ as recommended by Motorola, whilst the R/$\overline{\text{W}}$ needs no modification. The chip enable A, B generation

is unnecessary as the M6800 operates in eight bit mode, but it is included for completeness.

Figure 6.15 shows the logic required for a Z80 to Softy interface, where $\overline{VA}$ is defined as the presence of $\overline{MREQ}$ and $\overline{RD}$ or $\overline{WR}$. The direction indicators are included so that no data bus buffers are enabled until any external buffers are correctly enabled. This is necessary because of the short $\overline{WR}$ cycle of the Z80. As the $\overline{WR}$ cycle is now as long as the $\overline{VA}$, it can be used directly to act as R/$\overline{W}$. Again the sixteen bit decode is added although not required.

7. Master/Slave Interfacing

As stated in Chapter 4, the function of the interface board is the provision of a bidirectional direct memory access facility between two dissimilar microprocessor buses. This section will discuss the mechanism for providing this facility in more detail and will examine, in general terms, the circuit components and functional blocks required for the correct operation of such a facility.

There are several methods by which the necessary bus interface functions may be provided. Given that the two processors involved have a mechanism for releasing control of the bus to an external direct memory access controller, the two devices could directly drive the bus with no intervening buffers. This is shown in Figure 7.1. As shown, the two processors are of the same type, therefore there is no control signal conversion. Should two different processors be used in this way, the control signals would undergo transformation so that a common set of signals are transmitted to the system bus. The set of signals broadcast in this fashion may have the same type and timing of one of the processors or be a synthesis of the control signals of both. The criteria for selecting the system control signal type may be a desire to employ input/output devices designed for use with one processor, to obtain the benefits offered by the more complex control signal set or to reduce overall cost. Designs employing both microprocessors of the same family and from different families have found many uses, but with only one processor controlling the bus at once.

The bus arbitration circuitry is responsible for granting access to the processor with higher priority, where the priority may change on a regular basis, such as when interrupts occur, or when one processor requests services that can only be performed by the other processor. It may be felt that there is little to be gained by using two processors where only one may execute, but

this is not the case. Typically, if the processors are of the same type, one processor will check the results generated by the other so that, if a discrepancy occurs, corrective action may be taken. This system offers a form of multiple redundancy and, indeed, more than two processors may be involved. The Intel iAPX432 offers this testing ability as a design feature, one processor being nominated as bus master, the other as monitor. Should the monitor disagree with the bus master at any point, an exception is generated. The second form, where dissimilar processors are employed is usually adopted for one of two reasons. The different instruction sets of various processors give each a set of strengths and weaknesses. By combining two processors, each may execute program sections that represent a strength. This is seen most commonly with the arithmetic co-processors employed by many microcomputer families. The circuitry for floating point arithmetic is so complex that there is no room on the processor chip itself and it is placed in a separate package. Now when the main processor detects a floating point instruction, control is automatically passed to the floating point device until the instruction is complete. Latest processor designs so integrate these two processors that the main device will retain control of the address bus and perform all addressing mode calculation and open memory devices for access, whilst the co-processor controls the data bus and generates or accepts data directly.

The second occasion on which two dissimilar processors are employed is to allow users to continue running software from older microcomputer designs. This is a facility long offered by manufacturers of mainframe computers to persuade customers to change mainframes. The high cost of software development makes a new computer uneconomic unless programs already developed can be executed. Normally this facility has been offered by microcode or software emulation. With microcomputers, the cheapness of the processor chip itself has

led to the use of two hardware processors rather than a software emulation. This is most commonly seen with the CP/M operating system that will execute on the Intel 8080, 8085 and Zilog Z80 microprocessors. Currently, the largest body of microcomputer software is available for this system and, to retain the use of such software, newer sixteen bit based computer systems frequently contain one of the above processors. When execution of the CP/M operating system and programs is required, the sixteen bit processor halts and the eight bit processor gains bus mastery.

Where extra speed is required, it is possible that each processor will have a section of memory in which programs and data can be stored so that both (or all) processors may execute programs simultaneously. To allow the sharing of common resources, each must be buffered so that there is no bus conflict. This scheme, shown in Figure 7.2 presents few difficulties if the two processors are of the same type and have a method of waiting for slow memories. Should both processors attempt an access to the shared resource simultaneously, one (A) will be granted access and the appropriate data, address and control buffers will be enabled, the other (B) will wait, as it would for a slow memory device, until the first access is completed. The buffers of processor A will then be disabled, those of B enabled, andthe second access may now occur. As with the common system bus, if two processors of different type are employed, bus conversion will be necessary. If one, or either of the processors has no facility to wait for slow memory, the speed of the common memory must be such that, if two processors request an access, both may be satisfied in the time allowed for memory access. An arrangement capable of satisfying such a processing system is shown in Figure 7.3 Now the first processor to gain access presents an address and the memory responds in a fraction of the allowed time. The data is latched onto the local bus and the buffers of the processor closed. Whilst the second processor is accessing the

memory, the data bus latch is still presenting the information retrieved by the common memory to the processor. Thus, at the end of the cycle, the data on the bus is still valid and the cycle completes normally.

Although the above schemes demonstrate the facilities required for the system described in this thesis, none are suitable as they stand. The first system uses two processors sharing a common bus so that each may completely examine the memory of the other but, both may not run simultaneously and, as such processors are usually situated together, there is little difficulty in ensuring that all the necessary control signals are available. Also, as both potential bus masters are in the same location, the buffer control signals are generated more easily since the problem of direction of data during a read (or write) cycle does not exist - it is always towards (or away from) the processors. Whilst this makes the design more straightforward, the resultant dual processor circuit is both highly specialized and complex.

The second example has the two processors physically separate and capable of independent operation, but one processor cannot access the local memory of the other. This facility is essential if the supervising processor is to retrieve data from the slave without slave assistance. A major advantage offered by the use of common memory in multi-processor systems is as a communications area. If the two processors wish to transfer data, the common memory access, when a processor will only be waited if another processor is already using the memory, offers a low overhead means of transferring data.

For reasons already stated, the slave microprocessor is to operate within the confines of its own system with the bus protocol that is given by the processor generated control lines. As a result, any control signal transformation must be applied between the slave processor and the supervising processor. Similarly, any control signal transformation to be applied to the master processor for the purposes of interfacing to the slave processor should

not occur on the processor itself, unless it is a fixed modification that does not change with the movement between various slave microprocessors. If this is the case, boards that belong to the master system would also require alteration as the slave processor was changed.

The scheme thus arrived at is shown in Figure 7.4. Now each processor can make two types of request of the interface boards. The first is a request for an access to the common (or shared) memory. If that memory is not currently in use, the access request will be granted, the buffers opened and the relevant processor may read or write data. If the shared memory is in use by the other processor, the processor must wait for the access to be completed. The second type of request that can be made is for an access to the memory of the other processor. As both processors will normally be running, the bus will be in use and the appropriate request for a direct memory access transfer must be made of the processor. At some point this request will be honoured, whereupon both sets of buffers will be opened, allowing the address generated by one processor to be presented to the bus of the second and also allowing the data to be returned. Here the interface circuitry is responsible for converting the control signals generated by the direct memory accessing processor to those required by memory devices on the system being accessed.

The wide range of responses by processors to a request for direct memory accesses demands that care be taken in the precise sequence of granting a requesting processor access. For example, the Zilog Z80 will allow a direct memory access part way through the execution of an instruction, whilst the M6800 will only allow such an access at the end of an instruction. It is therefore necessary that the processor requesting direct memory access should not gain access to the shared memory in case the processor that is to allow a direct memory access will also require the shared memory before granting access to its own bus. It should also be apparent that if each processor

attempts to access the bus of the other at the same instant, neither will be able to grant access until the cycle has been completed. This leads to a 'deadly embrace'. Various methods exist for resolving this situation, should it occur, but few are satisfactory. If both processors attempt the access, the wait applied to one could be removed, without the buffers being enabled. This leads to the processor reading invalid data off the floating bus. Generating an interrupt of some kind can indicate an unsatisfactory bus access, but as each instruction may perform several bus accesses, the access which caused the failure cannot be identified, or the correct data restored. Newer processors offer a bus cycle abort facility that forces the processor to a section of program that deals specifically with such a problem, and, depending on the type of processor, the instruction may be retried, providing that no data within the processor has been affected. In all cases, it is preferable that software in the two machines co-ordinate accesses so that the situation does not occur. This can be achieved by ensuring that any accesses from the slave are limited to the shared memory or that, with co-operative software, a flag in shared memory is claimed by a processor before accessing the memory of the other.

It is also necessary that the supervising processor be able to completely control the action of the slave processor. Therefore the slave processors reset and interrupt lines should be available for examination or control. This must be done through an input/output device belonging to the master processor as, should it be a memory device, it might be possible for the slave to alter its conditions of operation. A further facility required is the ability to completely enable or disable slave accesses to the master. Should an access occur when not allowed the only available action is likely to be the halting of the slave processor and therefore the interface must be able to signal that such a halt has occurred, also providing a mechanism for the release of the

processor.

Given the cabinet arrangement of the system, it is possible to physically disconnect the supervising processor from the slave and a method of allowing electrical disconnection such that the slave processor activity is not disturbed should also be provided.

The main sections of an interface are therefore :-

i) a set of buffers that allows the interface to monitor the control signals of each processor and determine when an access requiring interface action is occurring.

ii) a set of buffers for the address and data buses of each processor that can be enabled as required for interface function.

iii) a wait state generator for each processor so that the access may be suspended until such time as the requested resource becomes available.

iv) circuitry to emulate the action of the processor being direct memory accessed and generating the correct bus control signals.

v) a section of memory available to either processor (the shared memory) and arbitration circuitry to control access.

vi) means to monitor and effect the reset and interrupt lines of the slave processor.

Having specified the general format and facilities required of an interface board to be used with a supervisory/slave system, Chapter 8 discusses in detail three such interfaces, between a Z80 and Motorola M6800 and M68000 and Intel 8086 microprocessors. In addition, the specific modifications to the basic interface layout required to allow the successful operation of the system with microprocessors having particularly complex bus structures are examined.

8. The Target Microcomputers

8.1 Specification

This work is based on the concept that a simple microcomputer with minimal facilities can be used as a demonstration system and can be studied providing that a larger microcomputer is available to assist in the interpretation and control of the demonstration microcomputer. For this to be effective, the target microcomputer must be as simple as possible and the complexity moved to the supportive machine. The chosen method of interfacing the two units using a bidirectional bus to bus interface offers several advantages, the most important of these being that no software need be located In the read only memory on the slave. On all the slave systems, EPROM is provided so that the boards may run programs when used as stand alone systems but provision must be made for the supportive microprocessor to perform direct memory access onto the slave microprocessor card. When this is to be done, random access memory must be at the reset location of that microprocessor so that the reset vector contents can be altered. However, when running as stand alone units, the reset vector must access EPROM memory. The address decode must therefore be alterable so that either type of memory can be positioned at the vector address.

For demonstration purposes each system should have available a parallel and serial I/O device from the support devices supplied to operate with that processor. A system that fulfils the above requirements is therefore suitable for use as a slave processor within this scheme.

The related interface for each processor must allow the supportive processor to reset, halt and interrupt the target machine and monitor these lines so that externally triggered changes to their state may be observed. The

supportive processor must be allowed to access any memory or input/output location available to the target system and, when permitted by the supportive processor, the target processor should be allowed to access the memory of the supportive processor, though not the I/O space as this would allow the slave to affect the conditions of its operation. Finally, the interface should offer a limited amount of memory that is available on a first come, first served basis to both processors so that neither will be delayed unless both access the memory simultaneously.

8.2 The M6800 CPU Board

The Motorola 6800 is an eight bit processor with a range of supportive devices that includes the M6850 Asynchronous Communications Interface Adapter (ACIA) and the M6821 Peripheral Interface Adapter (PIA) devices. On reset, the processor fetches a two byte program start vector from locations $FFFE_{16}$ $FFFF_{16}$. Arranged below the reset vectors are the Non-Maskable Interrupt (NMI), SoftWare Interrupt (SWI) and Interrupt ReQuest (IRQ) vectors ($FFFD_{16}$ to $FFF8_{16}$). The processor has an addressing mode that uses addresses 0000 to $00FF_{16}$ (the direct addressing mode) offering reduced program length and faster execution. It is therefore useful for M6800 systems to have EPROM at the $FFFF_{16}$ end of memory and random access memory at 0000 so that direct addressing can be used for variable storage. As there is no separate input/output address space, input/output devices are decoded as memory locations and any memory reference instruction and addressing mode combination can be used to access such devices. The address bus is validated by the $\phi 2$ line of the processor becoming high, as is data during processor write cycles.

There is no method of inserting extra clock periods in a bus cycle so that slow memory devices can be used. Instead, the entire clock must be slowed, a lower limit of one hundred kilohertz being imposed by the dynamic

nature of parts of the microprocessor. The complex timings required by the processor (two non-overlapping phases) coupled with the need to stretch clock cycles, make the use of a Motorola clock generator module (M6871A) extremely attractive to the designer.

Apart from the second clock phase ($\phi 2$) which acts as an address validator, there is a cycle invaildator (VMA) which indicates when an entire bus cycle is the result of an internal operation and should thus be ignored. VMA will become inactive when the cycle is invalid, but not between two valid cycles, even though the address bus at that point contains invalid values. It is therefore necessary to employ both $\phi 2$ and VMA in any address decoding scheme. All M6800 peripheral devices require a constantly maintained clock to allow the synchronization of internal operations and therefore require the availability of the $\phi 2$ signal which is accepted on the 'E' input.

The target M6800 system therefore employs a one megahertz M6800 microprocessor with the clock inputs driven by an M6871A clock module. All address and data lines leaving the microprocessor are buffered, the address lines by 74LS244 devices, the data lines by the bi-directional equivalent, the 74LS245. All these devices are disabled during direct memory access operations by the M6800 signal that indicates bus availability to external devices, Bus Available (BA). The data bus direction control is taken from the processor's R/$\overline{W}$ line. The buses are also buffered on the edge of the card, thus protecting all on board memory and input/output devices from noise induced on the backplane. The data bus is again buffered by a 74LS245, but the address buffers now also use these devices. This is to allow the master processor (or any other direct memory access device) to drive the address bus onto the card (a facility obviously not required *into* the microprocessor itself). The 74LS245 buffering the address bus are continuously enabled and the directional control is given by the BA signal. The data bus is only enabled when $\phi 2$ is high and a

data transfer is occurring across the board boundary. This will take place when the M6800 is reading from or writing to memory which is *not* on the microprocessor card, or when a direct memory access controller (primarily the supervisory processor) is accessing memory which *is* present on the microprocessor board. The directional control of the buffer is also altered in accordance with bus ownership. If the M6800 indicates that a write cycle is taking place, the buffer must drive away from the microprocessor. If an external device is writing, the buffer must drive towards the microprocessor. The derivation of this signal is shown in Figure 8.1. The address decoding for on board devices is based around a 74LS138 three to eight decoder. A four input NAND gate takes VMAO (a derivative of VMA that will be discussed later in this section) and the three high order address lines (A15 to A13). This generates a board select signal that decodes an eight kilobyte address space for devices on the microprocessor card. The 74LS138 and A10 to A12 are used to reduce this to eight, one kilobyte, spaces of which five are used. As mentioned previously, it is necessary that either EPROM or random access memory be located at the reset vector address. The address space containing the reset vector is decoded by the Y7 signal generated by the 74LS138. This signal, and the Y4 signal (decoding addresses $F000_{16}$ to $F3FF_{16}$) are fed to a change-over switch. The outputs of the switch are taken to an EPROM chip enable signal and a random access memory chip enable. Now, by reversing the switch, the random access memory will replace the EPROM and vice versa. In this way the supervisory microprocessor can insert new values into the reset and interrupt vector locations. The Y6 and Y5 signals output by the 74LS138 are taken to EPROM and random access memory devices respectively, giving a total memory capacity of two kilobytes of EPROM and two kilobytes of random access memory. The EPROM devices used are 2708, whilst each kilobyte of random access memory requires two 2114, one thousand and twenty four by four bit

memories. The chip enables of the EPROMs are also conditioned by the R/$\overline{W}$ signal so that it is not possible to enable the 2708s whilst any other device, such as the M6800 or supervisory microprocessor, is also driving the bus. The chip enable signals of the random access memory devices are not conditioned by the R/$\overline{W}$ line as it is used directly by these devices to control internal buffers.

The Y0 signal, decoding addresses $E000_{16}$ to $E3FF_{16}$, is used to select the on board input/output devices. The one kilobyte space resulting from the use of this signal is further reduced by the requirement that A6 to A9 must be high. The input/output devices consist of one M6820 PIA (an earlier version of the M6821) and one M6850 ACIA. Also provided are eight light emitting diodes which are available as a write only port. These LEDs, which are extinguished whenever the processor is reset, are driven by a 74LS273 latch and offer a simple method of allowing programs to indicate various conditions.

The address decode for these devices is partial and is shown below. In this table, 1 indicates that the address line must be high to select the device, 0 that the line must be low, U indicates that the device uses the address line whilst X indicates that the line is unused.

A5	A4	A3	A2	A1	A0	
X	X	1	1	U	U	PIA
X	1	X	X	1	U	ACIA
1	X	X	X	X	X	LEDs

As can be seen, it is possible to access several devices at once if addresses used to access the input/output devices are not carefully selected. For example, address $E3FF_{16}$ selects all the input/output devices simultaneously. Therefore the following addresses are used to access the various devices : $E3E0_{16}$ LEDs, $E3CC_{16}$ to $E3CF_{16}$ PIA, $E3D2_{16}$ to $E3D3_{16}$ ACIA. If the binary patterns for these addresses (as shown below) are examined, it will

be seen that no two devices are enabled simultaneously.

A5	A4	A3	A2	A1	A0	
1	0	0	0	0	0	LEDs
0	0	1	1	U	U	PIA
0	1	0	0	1	U	ACIA

The input/output lines generated by the peripheral interface adapter and the asynchronous communications interface adapter are taken to the bottom connector of the double eurocard so that connections to external experiments may take place. The pinout of the second connector is given in Table 4. In addition, an area of the double eurocard has holes drilled through it on a 0.1 inch matrix. This enables other integrated circuits to be added to the board by the use of wire wrap techniques. To facilitate connection to the PIA and ACIA, provision has been made for the connection of wire wrap pins to the outputs of these devices so that connection may be made into circuits placed in the wire wrap area. This area has been used to hold an LM555 astable, MC1488 and MC1489 RS232 interface devices, thus allowing the ACIA to drive RS232 devices directly. Any terminal connected to the ACIA can then be used to examine and alter the memory and registers of the M6800 system, under the control of a monitor program that resides in one 2708 EPROM that can gain control of the M6800 system after a reset (given that the EPROM is placed at the reset vector location by the change-over switch discussed earlier). It is thus possible to run the M6800 system completely independently of the supervisory processor if required.

The reset line of the M6800 microprocessor is designed for open collector operation and is therefore terminated with a 680 ohm pull up resistor. The signal is fed down the backplane via the eurocard bus connector and is also driven on board by the output of a 7403 open collector gate. This is provided with a resistor/capacitor timing circuit so that a power on reset occurs and a

push button switch is provided on the edge of the board so that control of the processor may be regained without removing power. The state of this line is continuously displayed by a green LED on the edge of the card towards the front of the rack. The LED is driven by two sections of the 7403 so that the LED is on when the reset line is at zero volts and hence the processor is in the reset state. Also monitored by an LED is the bus available line that indicates that the processor has halted or granted the bus to an external bus controller. This LED is red and is driven by one section of the 7403, being illuminated when the processor is in the halt state.

The three open collector lines, non-maskable interrupt ($\overline{NMI}$), interrupt request ($\overline{IRQ}$) and halt or bus request ($\overline{HALT}$) are provided with 680 ohm pull up resistors and the $\overline{IRQ}$ line is also taken to the peripheral and asynchronous communications interface adapters so that these devices may interrupt the processor if desired. If these devices are to interrupt, the $\overline{IRQ}$ outputs must be connected by wire links to the microprocessor $\overline{IRQ}$ line . As these lines are unbuffered, the current state of all these signals is available on the backplane and hence are available for monitoring by the supervisory processor via the interface card.

The M6871A clock module generates MOS ϕ1 and MOS ϕ2 specifically for the microprocessor and a TTL version of ϕ2 that is buffered and broadcast along the backplane. The M6871 requires two control signals, $\overline{HOLD1}$ which stretches the ϕ1 high period of the cycle, and MEMRDY which acts in the same fashion as $\overline{HOLD1}$ but for the ϕ2 high period. Output to the backplane for use by the interface to the supervisory microprocessor are the MEMCLK signal (an advanced version of ϕ2) and the 2. fc signal which is twice the frequency of ϕ1 and ϕ2 and is not held by either of the cycle stretching signals. It is thus suitable for timing the stretching of either phase.

The control signals R/$\overline{W}$ and VMA are buffered by sections of a 74LS125

quad three state buffer which is enabled by the M6800 BA signal, that is they are only broadcast by the M6800 when it has control of the bus. The BA and ϕ2TTL signals are also buffered by the 74LS125 but are permanently enabled.

As previously mentioned, the VMA signal is a cycle invalidator since it does not rise and fall with ϕ2. This is an anomalous signal and has been designed out of later versions of the M6800 family such as the M6809. The signal is not normally three state, but has been made so in this design by passing it through the 74LS125 buffer. Normally, when the bus is granted to an external device by the M6800, VMA is driven low so that external bus controllers cannot enable any devices using VMA in the address decode. In the present system VMAO is used in the address decode and this signal is generated by the M6800/supervisory microprocessor interface board. If this card is not present, VMAO will stay permanently high, thus removing the VMA signal from the decode. If the card is to be used independently of the supervisory microprocessor, provision is made to link VMAO to VMA so that invalid bus cycles are not allowed to access any memory or input/output devices.

8.3 The Motorola 6800 Interface

As both the supervising processor (Zilog Z80) and the Motorola 6800 are five volt devices with a TTL compatible bus with eight data and sixteen address lines, it might be thought that an interface card between these two systems is of low complexity. The bus structure of the Zilog Z80 follows the philosophy of the Intel 8080, both being similar to the processor shown in Figure 4.2. The Motorola M6800 is based upon the bus structure shown in Figure 4.1. Therefore the Z80 and the M6800 buses represent the earliest and least forgiving versions of two markedly different bus structures. The provision of voltage conversion is provided directly by various integrated circuit

facilities, whilst the problem of data bus width conversion and address bus manipulation is relatively simple to resolve, leaving the task of control signal conversion as the most complex undertaken by the interface circuits.

The general layout of the interface board is as shown in Figure 7.4. As each processor can perform direct memory access operations on the other, the address buses are buffered by 74LS245 drives. All sixteen of the M6800 address lines are made available to the Zilog Z80 bus when a direct memory access occurs. Note that the Zilog Z80 bus has twenty address lines in this implementation and the memory manager supplies a high order nibble of zero to the sixteen lines from the M6800. Under memory manager control the M6800 can therefore access any of the memory devices on the master system boards one and two. Of the twenty lines produced by the memory management unit on the Zilog Z80 system, the top four are discarded as they are not required and the bottom sixteen are available to the M6800 address bus during direct memory access, thus allowing the Zilog Z80 to access all memory and input/output devices attached to the M6800 system.

All the major signals of the Zilog Z80 are buffered, including those produced by the memory management unit. Thus, one 74LS244 buffers the unidirectional signals Zilog Z80 system clock (ϕ), instruction code fetch ($\overline{M1}$), input/output request ($\overline{IORQ}$), bus grant acknowledge ($\overline{BACK}$) and drives the interrupt priority chain line (IEI). The signals Other Micro Select ($\overline{OMS}$), shared memory request ($\overline{SMR}$) (both produced by the memory manager) and $\overline{IOSEL}$ which is used to decode the address for any input/output devices on the interface are also buffered. The outputs onto the interface board of all these signals are tied to the inactive state (high) by 2K2 resistors. Therefore if the buffer is disabled, the interface will be held in a stable condition equivalent to that which normally occurs when the Z80 is present but not accessing the interface (with one exception, the clock signal ϕ). A switch on

the interface controls the enable signal of this buffer and provides a means of isolating the interface logically before a physical disconnection. The Z80 may thus be used to set up a M6800 system and then be removed without interfering with the operation of the M6800. Three control signals of the Z80 must be driven by the interface during direct memory access operation, these are the memory request signal ($\overline{MREQ}$) and the read and write signals ($\overline{RD}$) and ($\overline{WR}$). In addition the input/output request line ($\overline{IORQ}$) is left floating during a direct memory access and provision must be made to drive this high (inactive). These requirements are met by an 74LS244 with the two halves operating during normal Z80 operation (bus signals onto interface card) and direct memory access operation (bus signals off card). The operation of these two sections are controlled by $\overline{BACK1}$ and BACK1 which are derived from the bus acknowledge of the Z80.

Similarly, the control signals of the M6800 are also buffered onto the interface board. One permanently enabled 74LS240 provides an inverted version of the two high order address lines ($\overline{A15}$) and ($\overline{A14}$). The first phase of the system clock ($\phi 2$) is inverted and effectively becomes $\phi 1$. The cycle invalidator (VMA) is also available in negated form as are the bus available signal ($\overline{BA}$) and the M6871A signal 2. fc and the $\overline{R}$/W signal. Buffered by three sections of a 74LS125 are the M6800 address lines A13, A12, A11. There is no direct equivalent of the $\overline{SMR}$ and $\overline{OMS}$ signals generated by the Z80 memory management system as this would require excessive customization of the M6800 system towards usage with a supervising processor. Instead, on board logic generates these signals from the address lines provided. Indeed, this is the reason for providing these address lines with a permanently enabled buffer as well as the controlled 74LS245 path mentioned previously. The M6800 equivalent signals of $\overline{SMR}$ and $\overline{OMS}$ are generated by a 10L8 PAL using the following formula :-

$$\overline{SMEM} = VMA \cdot \overline{BA} \cdot \overline{A15} \cdot A14 \cdot A13Etc$$

The A13Etc signal is a combination of the M6800 address lines A11 to A13 generated external to the PAL. From this it can be seen that an M6800 request for shared memory can only be generated under the three constraints that the address is valid, the M6800 has control of its bus and A15 to A11 contain the value 01111. This means that the shared memory appears in address 3800_{16} to $3FFF_{16}$.

Accesses to the Z80 are requested by the following formula :-

$$\overline{NA} = VMA \cdot \overline{BA} \cdot A15 \cdot A14 \cdot + VMA \cdot \overline{BA} \cdot \overline{A15} \cdot A14$$

Thus any valid memory reference from address 4000_{16} to $BFFF_{16}$ will attempt to access the Z80. Note the lack of a $\phi 2$ term in the above equation, which would normally be present. This is not required because the VMA employed throughout the interface board is not the address invalidator of the normal M6800 system, but an address validator. It is generated by the circuit shown in Figure 8.2. Reference to the M6800 data book[16] shows that the address bus becomes valid not more than 270 nanoseconds after the rising edge of $\phi 1$. In practice, as the address lines only have one input attached (that of the 74LS244 buffer), the address set-up time will be in the order of 100 nanoseconds, lower than the time allowed by this circuit which is 135 nanoseconds. This modified version of VMA is also used to generate the VMAO signal that is used in the M6800 system itself (Section 8.2). The equation of VMAO is given by :-

$$VMAO = VMA \cdot \overline{BA} + OMS \cdot BA$$

That is an address is valid when the M6800 controls the bus and the modified VMA is active or when the Z80 selects the M6800 and has control of the bus.

8.3.1 Wait State Generation

As stated previously it is necessary that either of the processors be

waited should the required resource not be available immediately. The Z80 circuit to achieve this involves the circuit shown in Figure 8. 3. As soon as a request is generated that involves any device on or beyond the interface card, the D input of the flip flop will be taken low. On the next falling edge of the Z80 system clock the low value will be passed to the open collector devices and thence onto the $\overline{WAIT}$ line of the Z80. The falling edge of the system clock is used since it is the point at which the processor samples the $\overline{WAIT}$ line. As there are delays, both to the interface card and from the interface card back to the processor card, the use of the falling edge assumes that the state of the wait line will always change at a point sufficiently far from the processor sampling instant so that the $\overline{WAIT}$ line will be in a stable state when tested. It should be noted that one wait state may be inserted via the memory management unit on the second board (Chapter 6). The system will now stay in this state as the request for the demanded resource is processed and the completion of this stage is indicated by the lowering of either shared memory available ($\overline{SMA}$) or target memory available ($\overline{TMA}$). Again, the falling edge of the Z80 system clock causes this signal to be passed through to the Q output of the wait state termination flip-flop, causing the preset input of the generation latch to be taken low. This state will be maintained until the Z80 removes the request at the end of the cycle.

The wait state circuitry responsible for holding the M6800 during shared memory or Z80 access is more involved in that there are two distinct clock phases to be treated. When the M6800 requests an access to either the shared memory or the Z80 bus, the $\overline{HOLD}$ signal from the 12L6 PAL is used to stretch the $\phi 1$ high time using the following equation :-

$$HOLD = (\phi 1 \cdot SMEM \cdot SMA) + (\phi 1 \cdot NA \cdot \overline{BACK})$$

The first term governs the access to the shared memory and the second access to the master bus. In each case, the elements are the $\phi 1$ signal itself,

the request for the device (SMEM, NA) and an active low indication of availability (SMA and $\overline{BACK}$). Thus, if the device requested is not available, a HOLD will be generated. If it is available, the ready term will become low and the hold will be removed. The MC6871A requires that the $\phi 1$ HOLD only be released when the $\phi 1$ signal would have been high under normal circumstances. In other words, all cycle stretches must be for whole $\phi 1$ periods. To achieve this, the circuit shown in Figure 8.4 is used. As soon as the $\overline{HOLD}$ line falls, the output of the flip-flop is taken low, applying the HOLD to the MC6871. When the hold signal is removed, the change will not be clocked through until MEMCLK is low when 2.fc is high. These two signals, although produced by the MC6871, are not affected by the HOLD signal. Thus the MEMCLK signal corresponds to $\phi 2$ and is thus low when $\phi 1$ is (or would be) high.

As the HOLD signal is synchronized to the M6800 clock, there will be periods when the HOLD signal is removed with the HOLD1 request still present. To overcome possible problems, the output of the latch Hold Acknowledge (HOLDA) is used on the interface board in preference to the hold signal itself.

When the Z80 accesses the M6800, the address (and data) buses will be stable on the interface board Z80 buffers by the time the bus becomes available. Then, as soon as the buffers are enabled, the signals will be passed onto the M6800 buses and can adequately meet the address set-up requirements of the M6800 bus without stretching $\phi 1$.

The stretching of the second phase of the M6800 system, $\phi 2$, is required in both directions (M6800 to Z80 and Z80 to M6800). This clock pulse indicates that data on the bus is valid and, due to the significant delays generated from the M6800 board to the Z80 board, additional time is required for the transfer. Two circuits are used for this purpose, one generates the delay when the M6800 accesses the Z80 (Figure 8.5) the other for Z80 accesses to the

M6800 (Figure 8.6).

Referring to Figure 8.5, the circuit consists primarily of a 74LS74. The first flip-flop acts as a simple divide by two on 2.fc. This signal generator is necessary as all other signals are stretched by the MEMORY READY signal and are thus not suitable for timing purposes. The OR and NOR gates control the start of the stretching process and, effectively, this occurs only during the ϕ1 high period of a request for the Z80 bus. The MC6871A requires the ϕ2 hold to occur prior to the start of the ϕ2 cycle involved. If the Z80 bus is not immediately available, the $\overline{\text{HOLDA}}$ signal is used to remove the ϕ2 stretch request which, as it operates directly on the preset inputs of the 74LS74, would otherwise occur simultaneously with the $\overline{\text{HOLD1}}$ signal. When the Z80 bus becomes available, the $\overline{\text{HOLDA}}$ signal is taken high, thus generating the ϕ2 stretch request. The flip-flop is held in its preset state until the end of the ϕ1 high time, and maintains the ϕ2 hold requests for three 2.fc periods after that time, thus adding approximately 675 nanoseconds to the Z80 access. This circuit is also used to hold off the generation of Z80 control signals.

When the Z80 accesses the M6800, the circuit of Figure 8.6 is employed. Once the M6800 bus becomes available, the Z26 signal is generated from OMS and BusAvailable, and is used to remove a preset signal which holds the circuit inactive under non-access circumstances. The ϕ1 signal is used to clock $\overline{\text{BA}}$ through the flip-flop, thus acting as a ϕ2 stretch request. This is held until the Z80 wait circuit releases the Z80 which will thus remove the OMS signal, forcing the preset input of the flip-flop to remove the hold. The output of the flip flop is also used to generate the $\overline{\text{TMA}}$ signal used by the Z80 wait state circuit to detect when the M6800 memory becomes available. This signal is first conditioned by the $\overline{\text{TMA1}}$ signal generated by the 10L8 PAL. This signal only becomes active at the end of the ϕ1 high period of a Z80 to M6800 access. The net effect of the circuit is that the ϕ2 time is stretched, by the Z80

wait state inserter, once it is released from the indefinite wait state. This lasts for the time taken by the Z80 to complete the bus cycle (two Z80 clock periods).

8.3.2 Bus Availability Detection and Buffer Enabling

The release of the processors from their respective wait states can only be allowed once the memory area required for the access is available. Broadly this memory is in one of three areas, the M6800 bus (Z80 accessing), the Z80 bus (M6800 accessing) and the shared memory (either processor accessing). In practice, it is easier to detect the availability of a processor bus since a signal is generated by the processor concerned. The shared memory does not, of itself, generate any signal and, moreover, is subject to simultaneous access requests.

When a request for a direct memory access cycle is made to the Z80, the $\overline{\text{BUSRQ}}$ signal is taken low. When all signals are tristate at the end of current bus cycle, the $\overline{\text{BUSAK}}$ signal is asserted. Theoretically, this signal would be sufficient to indicate that the slave could start a bus cycle. Should other devices be able to perform direct memory access, the presence of the $\overline{\text{BUSAK}}$ signal might indicate that the bus had been granted to some other unit. The correct solution to this problem is a bus request priority chain, with the lows being granted to the highest priority device requesting access. This requires the use of several connections throughout the system, which could not be accommodated with the connection format employed. An alternative technique, where software assumes that only one device possesses direct memory access capability at any one time, is employed. Here the processor grants authority to one device only and, having obtained direct memory access permission, any $\overline{\text{BUSAK}}$ generated is thus aimed at that device. It is necessary that when authority has not been granted, no response to a $\overline{\text{BUSAK}}$ cycle be made. The interface board has the ability to enable or disable slave access to the Z80

system and is therefore suitable for use in the suggested manner. An implication of the scheme is that the $\overline{BUSAK}$ signal must be conditioned in some manner so that $\overline{BUSAK}$ is only seen by the interface hardware when requested. This is achieved with the circuit of Figure 8.7.

The $\overline{BUSRQ}$ signal can only be generated when the M6800 requests an access ($\overline{NA}$) and the interface board currently has direct memory access authority controlled by the Z80 available signal ($\overline{ZAV}$). Once the $\overline{BUSAK}$ signal is generated in response to the presence of both these signals, BACK1 is presented to the board, but only for the duration of the $\overline{BUSRQ}$ signal.

In similar fashion, the Z26 signal is produced to indicate that a Z80 access of the M6800 is proceeding whilst the TMA signal indicates that the bus is available and $\phi 2$ is high. The shared memory is allocated with the SMA signal which is generated by the 12L6 PAL using the following equation :-

$$SMA = (\overline{SMEM} \cdot SMR) + (SMA \cdot SMR)$$

When the SMA = 1 the Z80 has access to the shared memory and when SMA = 0, the M6800 has access.

The bus buffers must be enabled at the appropriate point for the access currently undertaken. The address buffers between the Z80 and the interface board are enabled as soon as the requested resource becomes available. This is indicated by the $\overline{BACK1}$ signal for M6800 to Z80 accesses and the $\overline{TMA}$ and $\overline{SMA}$ signals for Z80 to M6800 or shared memory accesses. The direction of these buffers is decided by the $\overline{BACK1}$ signal. When the Z80 is in control, the buffers drive to the interface, and when the M6800 is in control they drive to the Z80. The data bus buffer enable is similarly treated, with the exception that it is disabled when a request is made to the on board input/output devices which have a separate buffer. The direction of the data bus buffer is given by the exclusive OR of the $\overline{BACK1}$ signal and the derived R/$\overline{W}$ signal R/$\overline{W}$(SMEM), which is produced from the Read/Write line of whichever

processor currently controls the interface card. The data bus buffer for the I/O devices is permanently enabled, but only drives towards the Z80 (as opposed to towards the I/O devices) when the Z80 reads the on board I/O locations.

The address buffers between the M6800 and the interface card are controlled in a similar fashion. The $\overline{E1}$ signal is generated by the 12L6 PAL using the following equation :-

$$E1 = TMA + (NA \cdot BACK) + (SMEM \cdot \overline{\phi T} \cdot \overline{SMA})$$

These three terms indicate that the M6800 is available (Z80 accessing) or that the Z80 is available (M6800 accessing) or that the shared memory is available (M6800 accessing). The direction is given by the M6800 analogue of $\overline{BACK1}$, $\overline{Z26}$ which has previously been described. Likewise, the data bus enabling circuit is similar to that of the Z80 data bus buffers, without the complications of the second I/O data bus buffer. Here the buffer is enabled by E1 and direction is given by TMA exclusively ORed with the R/$\overline{W}$(SMEM) signal mentioned previously

8.3.3 Control Signal Generation.

When access has been requested and the appropriate device becomes available, there are two signals that must be generated, the address strobe (or, for the shared memory, the chip enable) and the data flow control signal ($\overline{RD}$, $\overline{WR}$ or R/$\overline{W}$). This task is simplified for accesses into the M6800 as the address validation is inherent in the bus availability signal, in that it is established by the two clock pulses, and thus controls rather than is generated. The R/$\overline{W}$ generation required is performed simply by gating the Z80 $\overline{MREQ}$ and $\overline{WR}$ signals through a tristate buffer enabled when the M6800 bus is available. Thus, when the Z80 performs a write cycle into the M6800 cycle, R/$\overline{W}$ adequately meets the set-up specification, being available from the start of $\phi 1$.

For M6800 accesses into the Z80 $\overline{MREQ}$, $\overline{RD}$ and $\overline{WR}$ must be generated. All these signals are produced by the 10L8 PAL using the following formulae :-

$$\overline{MREQ} = VMA1 \cdot \overline{BA} \cdot \overline{\phi T}$$

$$\overline{RD1} = VMA1 \cdot \overline{BA} \cdot \overline{\phi T} \cdot RW$$

$$\overline{WR1} = VMA1 \cdot \overline{BA} \cdot \phi 1 \cdot \overline{RW}$$

MREQ1 and WR1 signals are also conditioned by the $\phi 2$ clock stretching circuitry discussed previously, so that they are only present for part of the M6800 cycle, thus ensuring that both data and address are stable for the entire cycle. The three signals are then gated onto the Z80 control bus by part of a 74LS244 enabled by $\overline{BACK1}$.

The shared memory accepts the R/$\overline{W}$(SMEM) signal, also generated by the 10L8 PAL with the following formula :-

$$\overline{R}/W(SMEM) = WR \cdot SMA + \overline{RW} \cdot \overline{SMA}$$

The first term generates a high when the Z80 owns the shared memory and is writing. The second is high when the M6800 owns the shared memory and is writing. Thus R/$\overline{W}$(SMEM) goes low whenever the owner of the shared memory is writing.

The shared memory is provided by two, one kilobyte blocks, each with its own chip enable. The respective chip enable signals are produced by the 12L6 PAL using the following formulae :-

$$CE1 = A10 \cdot BUFF \cdot [(\overline{\phi T} \cdot SMEM \cdot \overline{SMA}) + (MREQ \cdot SMR \cdot SMA)]$$

$$CE2 = \overline{A10} \cdot BUFF \cdot [(\overline{\phi T} \cdot SMEM \cdot \overline{SMA}) + (MREQ \cdot SMR \cdot SMA)]$$

The terms common to both equations, in square brackets, are the address valid, shared memory request and grant signals for the M6800 and Z80 respectively. The A10 term selects which device is to respond. The BUFF term is used to delay the generation of the chip enable until the appropriate address buffers have been enabled, thus allowing the A10 signal to be available to correctly select the appropriate device.

8.3.4 M6800 Static Control

It has already been stated that the Z80 must be able to influence the state of the M6800. Most of this function is performed by the on board parallel input/output device. Four output lines can generate reset, halt, non-maskable or maskable interrupts in the M6800 system, the PIO being the only circuit that can generate a reset condition other than the front panel switch. The PIO drives these open collector lines through the appropriate type of buffer and the state of the four mentioned lines can be monitored (BA rather than halt as it indicates a processor halt, rather than the request for a halt) by the Z80 since four PIO input lines are attached to the lines as they join the M6800 backplane. This is a useful facility as, in most cases, the PIO is not the only possible source of an active signal. For example, M6800 input/output devices could generate an interrupt and by monitoring the lines, the PIO can generate an interrupt to the Z80 so that any special action required can be performed. The PIO also produces the Z80 AVailable signal (ZAV), which is used to allow or disable accesses by the M6800 to the Z80. Should the M6800 attempt to access the Z80 when it has not been given authority, an illegal access attempt signal (IAA) will be generated by the 12L6 PAL using the formula:

$$IAA = (NA.\overline{ZAV}) + (\overline{ZAV}.IAA)$$

The second term in the equation latches IAA so that once generated, IAA will remain active until released by the Z80. The releasing of IAA requires that the Z80 temporarily grants access and this would normally be performed with the M6800 halted. The IAA signal is combined with the halt line from the PIO so that in the event of an illegal access attempt, the M6800 is halted. IAA is also fed into the PIO as an input and also drives a light emitting diode to give a visual indication of the cause of a halt condition.

In total there are four possible sources of a halt request: an attempt by

the Z80 to access the M6800, the PIO, an illegal attempt by the M6800 to access the Z80 (IAA) and the counter system used to single step the M6800.

The requirement that the M6800 should be single stepped is met by the provision of a count down circuit clocked by the M6800 system clock ($\phi 2$). The circuit consists of two 74LS161A counters whose load inputs are connected to the interface board input/output bus. Decoded as an input/output device, the counters are loaded with a preset value whenever the Z80 writes to location 34_{16}. The circuit is designed so that, when the counters (which count up) count from FF_{16} to 00, they halt and generate from the ripple carry output, a low going signal that is used to enable part of a 74LS139 one of four decoder. The select inputs of this device are provided by the PIO and are so organized that the resultant low produced by the LS139 can be fed into the M6800 halt, non-maskable interrupt, maskable interrupt or the Z80 PIO to cause a Z80 interrupt. The enable signal (T) of the counters is also provided by the PIO, from the same port that controls the halt and interrupt lines. This allows software on the Z80 to halt the M6800, load a value into the counter circuit and, with a single output instruction, both release the M6800 from the halt state and start the counter. As the counter is operated by the M6800 clock, the effect of loading the counter and releasing it is to fix the number of M6800 bus cycles that will occur before the event selected by the PIO occurs. For single stepping routines, provided by interrupts, the M6800 would initially be halted with the interrupt vector pointing to a small routine responsible for placing the current value of the stack pointer in a predetermined location and then executing a wait for interrupt instruction. As the M6800 stacks all registers as part of the interrupt response, all register values can be examined, or altered, once the value of the stack pointer is made available to the Z80. The Z80 would now load the counter with a value that will allow the M6800 to start executing the first instruction when

released from the halt state, whereupon an interrupt will be generated. The interrupt will only be recognized at the end of the instruction. Once the M6800 has executed the wait for interrupt instruction, examination of the registers can proceed. To allow the M6800 to execute the next instruction, another interrupt that causes the stack pointer to be reloaded from the fixed location must be generated, followed by a timed halt that returns the sequence to the start.

8.4. The Motorola M68000

The Motorola M68000 is a microprocessor capable of handling thirty-two bit data with a sixteen bit external data bus. Whilst initially relying on Motorola M6800 peripheral devices for support, it now has a range of more powerful devices specifically matched to its abilities. On reset, the processor loads a thirty-two bit value from locations 0 to 3 which is used as the initial value of the stack pointer, whilst the program counter is loaded from locations 4 to 7. In total, the first kilobyte of memory is reserved for vectors, providing two hundred and fifty five vectors, each of thirty-two bits. The addresses and function assigned to each vector can be found in [17].

Only one addressing mode differentiates between areas of memory and that is the absolute addressing mode, which can access either the entire address space (eight megawords each of sixteen bits) or those memory locations which lie in the bottom sixty four kilobytes of the address space (absolute short mode).

Because both vectors and the absolute short mode operate on the lower part of the address map, it is usual to place memory devices in the map starting at zero. As there is no differentiation made between memory and input/output devices in respect of the addresses they may occupy, input/output devices can be placed as desired.

The bus interface of the M68000 shows major improvements over that of the

M6800, particularly the ease of accommodating slow memory devices and the clear indication of when data and addresses are stable and valid.

The processor can access a memory organized as eight megawords. This implies a twenty three bit address bus and this is provided by the signals A1 to A23. As the processor can perform byte accesses, the bottom address line that would normally be used to select a byte address has been replaced by two signals that select the upper or lower bytes, upper data strobe and lower data strobe ($\overline{UDS}$ and $\overline{LDS}$).

The value on lines A1 to A23 is validated by the address strobe signal ($\overline{AS}$), whilst data for write operations is validated by $\overline{UDS}$ or $\overline{LDS}$. The validation of data during a read operation is performed by an external device asserting $\overline{DTACK}$ (Data Transfer ACKnowledge). This signal is also used to allow the processor to be interfaced to slow memory devices since, until this signal is asserted, the processor will wait indefinitely with address (and in write cycles, data) stable.

A problem arises should the processor attempt to access a location which has no memory devices associated with it, since, in such circumstances, the processor would wait with no means of recovery. To protect systems against such circumstances, the Bus ERRor input ($\overline{BERR}$) will force the processor to abandon the current cycle. The precise operation of the $\overline{DTACK}$ and $\overline{BERR}$ signals is left to the system designer and there are fundamentally two methods of designing the circuitry to drive these inputs. In the simpler case, whenever the processor asserts the address strobe signal, a delay is initiated corresponding to the time taken to access the slowest device in the system plus a suitable safety margin. Should the memory address decode circuitry not generate $\overline{DTACK}$ within the period of this delay, $\overline{BERR}$ is asserted. If $\overline{DTACK}$ is generated, it resets the delay circuitry so that an bus error is not generated. The second possible scheme is to automatically generate $\overline{DTACK}$ so

that it will occur at a preset time. This time period can then be influenced in one of two ways. Either a wait request may be submitted to the circuit so that the $\overline{\text{DTACK}}$ signal is delayed or a $\overline{\text{BERR}}$ is generated before the $\overline{\text{DTACK}}$ signal, thus terminating the cycle.

The circuit used in the present design is a synthesis of the two described above and is more suitable for use in a system where the M68000 is accessing another processor's memory space. The processor card generates a $\overline{\text{DTACK}}$ signal a preset number of clock periods after either $\overline{\text{UDS}}$ or $\overline{\text{LDS}}$ have fallen, indicating that data is to be transferred. This signal is only produced for the range of addresses corresponding to devices on the processor or interface cards or in the supervisory system. To ease the design of the circuitry, the entire lower sixty four kilobytes of the M68000 address map is considered as being inhabited. The circuit employed is shown in Figure 8.8. The falling edge of either the $\overline{\text{UDS}}$ or $\overline{\text{LDS}}$ signals releases a shift register composed of a 74LS174, and also places a high level on the shift register input. The register is clocked from the 68000 clock signal ϕ which is gated with the $\overline{\text{WAIT}}$ line from the edge connector such that, if $\overline{\text{WAIT}}$ is asserted, the clocking pulses do not reach the shift register. Dependent on the number of stages of the 74LS174 used, the high level will appear at Q4 several clock cycles later and, if the address decode circuit indicates that the bottom sixty four kilobytes of memory has been selected (A16 to A23 all zero), $\overline{\text{DTACK}}$ will be passed to the processor. If a $\overline{\text{DTACK}}$ signal has not been generated, two cycles later, either by the on board logic or external circuitry, a bus error condition will be signalled to the processor.

External circuitry has several options available. Firstly, if the address lies in the lower sixty four kilobytes, to take no action and allow the generation of $\overline{\text{DTACK}}$. Secondly, to use the $\overline{\text{WAIT}}$ line to delay the production of $\overline{\text{DTACK}}$, thus producing the full $\overline{\text{DTACK}}$ delay after the release of the $\overline{\text{WAIT}}$ line.

Finally, to generate its own $\overline{DTACK}$ and $\overline{BERR}$ signals. These options are available at all the addresses and, indeed, either $\overline{DTACK}$ must be supplied or $\overline{BERR}$ will be generated by default. The final option for the lower sixty four kilobytes is to allow the generation of $\overline{DTACK}$, but to inhibit the signal before it is passed to the processor. This can be accomplished by removing the DTEN signal, which, when low, inhibits $\overline{DTACK}$ at the output of the delay circuit, thus causing the prompt generation of $\overline{DTACK}$ once DTEN returns to a high logic level. A major use of the DTEN signal is to allow a single cycle circuit to be implemented, holding each bus cycle stable for examination.

The bus structure of the M68000 has no requirement for the $\phi 2$ and VMA signals of the M6800, but they are specifically generated by the processor to allow the interfacing of M6800 peripheral devices to the M68000.

As processor bus accesses are no longer synchronized to the clock input, clock generation is simpler and, typically, consists of a simple crystal controlled oscillator. On the processor board in this scheme, an inverter is used to drive the crystal which supplies a four megahertz clock signal to the processor.

Buffering is provided in a similar fashion to that described for the Motorola M6800 system. The twenty three address lines are buffered as they leave the processor by three 74LS244 devices which are disabled only when the M68000 passes control to another bus master, this being indicated by the bus grant acknowledge signal to the processor. The data bus is buffered by two 74LS245 devices, again disabled by bus grant acknowledge and, as the 74LS245s are bidirectional, the R/$\overline{W}$ line is used to select the appropriate direction. As the address and data buses leave the processor card, they are again buffered to reduce the effects on the on board memory devices of noise on the backplane. Here the address buffers are also 74LS245 devices to allow the master system access to the on board memory. The direction of the buffers is

controlled by bus grant acknowledge, the buffers driving on board when an external bus master is active. The buffers are enabled either during direct memory access or when the M68000 is accessing off board devices. Although twenty three lines are buffered, only A1 to A19 are presented to the backplane due to pinout restrictions.

The sixteen data lines are also buffered with 74LS245s, the enable signal being that for the address buffers, whilst the direction of the buffers is controlled by exclusivly ORing the R/$\overline{W}$ signal and the bus grant acknowledge signal.

All the devices on the processor card lie in the address range of 0 to $3FFF_{16}$, the map being given in Table 5. The major part of the address decoder is a 74LS138 decoder, enabled only when the $\overline{AS}$ signal and address lines A23 to A14 are zero. Lines A13, A12 and A11 are used to select different memory devices on the board. The 74LS138 generates a word enable signal, that is, it indicates that a particular sixteen bit location is being accessed. Upper data strobe and lower data strobe are then used to condition the selection signals provided by the 74LS138. The on board memory consists of two kilowords of random access memory (eight 2114 devices) and two kilowords of EPROM (four 2708 devices). The board provides an image of either a RAM or a ROM at the reset vector location, dependent on the position of a switch mounted on the board. The devices are deliberately imaged to the reset location so that they can always be accessed at their normal memory address regardless of the state of the switch.

Input/output devices are selected by a second 74LS138 cascaded from the first and each of the enable signals produced by this device corresponds to sixteen words. Of the eight enables provided, three are used. A 74LS273 latch is used to drive eight LEDs that can be used as software status indicators, whilst one Motorola M6821 Peripheral Interface Adapter and one M6850

Asynchronous Communications Interface Adapter provide parallel and serial interface capability. The two devices require that the timing of the bus meets M6800 standards, and this is assured as the selection of the input/output devices forces the generation of the Valid Peripheral Address ($\overline{VPA}$) signal. When this is presented to the M68000, the bus cycle is stretched until the next E signal high time, when $\overline{VMA}$ will be asserted and the transfer to the M6800 peripheral will take place.

All the above devices use an eight bit wide data bus, so they are connected to D0 to D7, thus appearing at the even word addresses.

The normal interrupt mechanism of the M68000 is based upon the interrupting device supplying a vector to the processor during an interrupt acknowledge cycle. As M6800 peripheral devices do not have this capability, an alternative mechanism, autovectoring, is provided. To allow the use of autovectoring, an interrupting device must assert the $\overline{VPA}$ signal during the interrupt acknowledge. Of the seven different levels of interrupt available, four will generate the $\overline{VPA}$ signal and the interrupt outputs of the on board M6800 peripherals are connected to these lines, other inputs being free for vectored operation.

When power is first applied to the M68000, both the $\overline{RESET}$ and the $\overline{HALT}$ lines must be held low for one hundred milliseconds to ensure a full processor reset. Thereafter, the processor can be reset by holding these lines low for ten clock cycles. The power on reset is achieved by an RC network, whilst a reset may also be generated by the operation of a push button, or the driving of a backplane line. All these signals are fed onto the $\overline{HALT}$ and $\overline{RESET}$ lines by open collector gates.

Both the reset and halt signals on the M68000 are bidirectional. The reset input is used to bring the processor to a known state, whilst as an output it allows software to reset all circuitry connected to the reset line

by the execution of a RESET instruction, which does not affect the internal state of the processor. Should the processor be required to stop execution, this can be achieved by driving the $\overline{\text{HALT}}$ line low and, in the event, that the M68000 cannot continue executing instructions due to bus errors occurring during the bus error processing sequence, this line will be driven by the processor. The $\overline{\text{HALT}}$ line has a further function since when a bus error is signalled, if $\overline{\text{HALT}}$ is taken low, the processor will abort the cycle and repeat it immediately. This allows the processor to access memory devices at a lower priority than an external device and to be forced off during the bus cycle without an excessive software overhead on the cycle retry.

The dual nature of both the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ lines leads to design difficulty in that, if the line is to drive other devices, it must be buffered, yet, if it is buffered, external devices cannot drive the processor pin as desired. This is best resolved by providing two lines for each signal, an inward open collector line and a buffered output with devices connected to the appropriate version. Due to pin out limitations, this approach was not adopted and the $\overline{\text{HALT}}$ line is open collector and only buffered when fed into an LED, whilst the bidirectional capability of the $\overline{\text{RESET}}$ line has not been utilized, the signal passing down the backplane being affected by the manual reset switch and external drivers only.

8.5 The Motorola M68000 Interface

The Z80 request detection circuit for this board can be split into two sections. The first section is responsible for claiming the interface board for either shared memory or slave accesses. As such, it detects the presence of $\overline{\text{SMR}}$ or $\overline{\text{OMS}}$ as provided by the memory management unit and the resultant signal is fed into the arbitration circuit. The second recognition circuit provides a signal to the direct memory access request generator whenever $\overline{\text{OMS}}$ is active. Detection of M68000 requests is more involved in that the address

decoding must be performed first. This is achieved by a 7425 dual four input NOR gate which is attached to the M68000 address lines A19 to A16 and A15, A13, A12 and A11. The address strobe signal is used to ensure that only valid memory addresses in the range 0 to $0FFFE_{16}$ generate a request. A15 is used to reduce the range further, to addresses between 08000_{16} and $0FFFE_{16}$, a range of thirty two kilobytes. This address range is used to request a Z80 memory access. The shared memory is only requested when A14 is high, and all other signals fed to the NOR gates are low, resulting in a request being generated in the range 04000_{16} to $047FE_{16}$. Whenever either of these signals becomes active, an M68000 request is fed to the shared memory request arbitration circuit. Before an access to either the shared memory or the other processor system can proceed, the requesting processor must claim control of the interface board. This is achieved by both requests being fed into the arbitrator which consists of two cross coupled NOR gates. When an input is taken low, by either the Z80 or the M68000 requesting the board, the gates will allow through only one of the requests whilst the other will be blocked. This result in either the activation of $\overline{A}$ (when low the Z80 owns the board) or $\overline{B}$ (when low the M68000 owns the board).

In the case of both the Z80 and the M68000, the presence of a request involving the board without ownership being granted, as indicated by $\overline{A}$ or $\overline{B}$ causes a wait signal to be passed to the relevant processor. As soon as the appropriate ownership signal is active, the wait request is removed from the processor and the access proceeds. The possibility of a deadly embrace when none should occur can be caused by the shared memory arbitration circuitry being activated when the other processor is being requested. For example, in a cycle in which the Z80 requests the M68000, it will request, and gain ownership of the shared memory. If the M68000 is also requesting the shared memory, it will wait until the Z80 access has finished. The correct action

would be to request the M68000 bus, and, once this is granted, ownership of the shared memory is guaranteed. Under these circumstances, the shared memory would be available to allow the M68000 to complete the current bus access.

Apart from the immediate wait states forced on the processors by an unsatisfied access, provision must be made to continue the wait states after the facility requested has become available to ensure that the timing requirements of the selected devices are met. In the case of the M68000, part of this requirement is met by the $\overline{\text{DTACK}}$ generator on the processor card which, when waited, will hold the processor at the *start* of the cycle. Thus, when the wait signal is removed, a full bus cycle period will occur before the cycle ends. It is therefore only necessary that the M68000 is held for the interval between gaining the interface card (which releases the M68000) and the point at which the Z80 bus becomes available. This is achieved by combining the signal indicating an access to the Z80 ($\overline{\text{DMAREQ}}$) with the Z80 bus availability indicator ($\overline{\text{BACK}}$). In this way, the M68000 is held from the initial request until the point at which it is driving the Z80 bus, at which point it will be allowed to complete the access.

The Z80 is similarly held until the M68000 has released its buses, although the circuit required is more complex since, to improve response to direct memory access requests, the M68000 acknowledges the bus request *during* the last bus cycle. It is therefore necessary for the interface board to complete the handshake operation with the M68000 before driving the M68000 bus.

The sequence for a second bus master claiming the M68000 bus is based around the signals Bus Request ($\overline{\text{BR}}$), Bus Grant ($\overline{\text{BG}}$), which is provided by the processor, and Bus Grant ACKnowledge ($\overline{\text{BGACK}}$), sent by the requesting device to the processor.

When the processor is active, the device requesting a direct memory

transfer (in this case the interface board and Z80) asserts the bus request signal. Several processor clock cycles later the processor will assert the bus grant signal. This does not indicate that the bus is available, but that it will become available with the removal of the address strobe signal. As soon as the bus is available, the external device may assert the $\overline{BGACK}$ signal to indicate that it now has control of the bus whilst, at the same time, removing the $\overline{BR}$ signal to prevent the M68000 re-entering the bus arbitration sequence. At this point the external device has control of the bus which it will retain until it releases the $\overline{BGACK}$ signal.

As soon as the address strobe and bus grant signals indicate that the processor has released the bus, a delay of two clock cycles must be inserted so that M68000 bus buffers have ample time in which to become high impedance and to ensure that the cycle time specification of memory devices will be met.

The circuitry that accomplishes the above protocol is centred around IC A48 of Figure 8.9. Upon the generation of the other micro select ($\overline{OMS}$) signal by the Z80 memory management unit, the arbitration circuit will eventually give control of the interface card to the Z80 (indicated by $\overline{A}$ becoming zero). This generates a high on one input of NAND gate D76b, which already has a high provided by the inverted output of the cleared SR latch, which is held in this state between Z80 accesses of the M68000. Bus request is therefore asserted and, when bus grant and address strobe indicate that the bus is free, AND gate D5a provides a high input to the NAND gate attached to the $\overline{S}$ input of the SR latch. This in turn asserts $\overline{BGACK}$ and removes the bus request signal to the M68000. As the only device capable of performing direct memory access in the M68000 system is the interface board, no further arbitration is required and the generation of $\overline{BGACK}$ indicates that the interface now controls the bus. Two D type flip-flops are freed from the clear state by the generation of $\overline{BGACK}$, as is the gated Z80 clock signal. The two flip-flops thus provide a delay of

two Z80 clock periods. At the end of this delay, the Z80 $\overline{WAIT}$ signal is removed.

As stated in the discussion of the M68000 processor board, only address lines A1 to A19 are used on the backplane due to pinout constraints. The M68000 address lines are buffered by three 74LS245 devices, the directional control for these buffers being provided by the interface generated BGACK signal, such that when the M68000 has control, the buffers drive towards the interface card. The buffers are enabled by a signal from the arbitration circuit which becomes active whenever the interface board is required to act. This signal is further conditioned so that if the Z80 is accessing shared memory the buffers are explicitly disabled. This results in the buffers being enabled for Z80 to M68000, M68000 to shared memory or M68000 to Z80 accesses only. A similar scheme controls the Z80 address buffers. The directional control is provided by the Z80 bus acknowledge signal and the buffers are enabled by the same arbitration produced signal used to enable the M68000 address buffers, but here combined with the M68000 shared memory access signal.

The data bus of the M68000 is sixteen bits wide, whilst that of the Z80 is eight bits wide. The interface board circuitry is responsible for reconciling this disparity for all M68000 to Z80 accessing modes.

The Z80 has two types of memory access, these being the normal eight bit data read/write cycle and the shorter instruction fetch cycle. In both cases the cycle length will be affected by any active wait circuitry, so that only the simple eight bit access is apparent to the interface. The M68000 can access data either as a byte from the lower or upper data buses, or alternatively, as sixteen bits of data from both sections of the data bus. Indeed, for instruction fetch operations, the sixteen bit transfer is mandatory. As the processor will allow eight bit accesses, all memory and

input/output devices in the system must also allow eight bit accesses. In this way, the Z80 can utilize only eight bit accesses and yet place any data required in the M68000 system. Should the M68000 attempt a sixteen bit access on the eight bit Z80 bus, additional circuitry will be required to perform two eight bit data transfers and sequence the data bus buffers in the correct order.

The data paths across the interface are shown in Figure 8.10 and it can be seen that only one eight bit latch is required to allow all possible modes of data transfer across the interface.

Examining first Z80 accesses either to the M68000 or the word organized shared memory, A0 from the Z80 (for which there is no direct equivalent in the M68000 system) is used to select the data path leading to either D0 to D7 *or* D8 to D15. There is a direct relationship between A0 and the byte selection signals of the M68000 as follows : when A0 is low, $\overline{UDS}$ should be asserted, when A0 is high, $\overline{LDS}$ should be asserted. The direction of all four data bus 74LS245s is determined by the R/$\overline{W}$ signal which is provided directly by the M68000 when it gains control of the interface board or, when the Z80 controls the interface, by an inverted Z80 $\overline{WR}$ signal. Thus, whenever the on board R/$\overline{W}$ signal is high, the data direction is away from the Z80 and towards the M6800. When the R/$\overline{W}$ signal is low, the reverse applies. The data buffers between the interface board and the M68000 are controlled by the $\overline{UDS}$ and $\overline{LDS}$ signals. Whenever either of the byte transfer signals are active, the appropriate data buffer is opened. The $\overline{UDS}$ and $\overline{LDS}$ signals will also be generated by the Z80 when accessing the word organized shared memory and therefore both these enable signals are conditioned such that the buffer remains disabled when the Z80 is accessing the shared memory, implementation of the disable function being identical to that for the M68000 address buffers.

When the M68000 accesses the Z80, with a byte operation required, the

74LS373 latch is disabled and the four 74LS245 buffers are controlled directly by the $\overline{UDS}$ and $\overline{LDS}$ signals.

When the M68000 performs a word read access on the Z80, the word address is placed on the bus, $\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$ and $\overline{LDS}$ are issued. The access is controlled by the circuit of Figure 8.11. With these signals present, $\overline{BUSACK}$ from the Z80 the Q output of the SR latch IC A48b goes high, negating DTEN, thus disabling $\overline{DTACK}$ to the processor. The M68000 will therefore wait at the end of the bus access cycle, by which time the address passed to the Z80 data bus will have generated the first eight bits of data. The 74LS373 latch is enabled via IC D20d. Since DTEN is low, the M68000 will continue to wait, but $\overline{DTACK}$ will be transmitted through the system from the on board $\overline{DTACK}$ generation circuitry, unaffected by the lowering of DTEN. When this signal arrives on the interface board, it enables the M68000 clock onto IC A36. The passage of high data through this shift register causes the following actions to be taken :

After one clock period, the input of the 74LS373 latch is disabled via Q1 and IC D20d.

After two clock periods, $\overline{AS}$ is taken high by the action of OR gate E5d, fed by D12c.

After three clock periods, A0 is taken to one by Q3 and the OR gate E5c.

After four clock periods, $\overline{AS}$ is again taken low, thus causing a read to be performed to obtain the next data byte. The lower data bus buffer is disabled by the NAND gate C42b

After six clock periods, $\overline{Q6}$ and $\overline{WREN}$ enable the 74LS373 output, disable the upper data bus buffer and, as NAND gate 42b goes low, the lower data bus buffer is also enabled, thus presenting sixteen bits of data to the M68000. The output Q6 is used to reset the SR latch, thus allowing DTEN to rise.

The M68000 now receives $\overline{DTACK}$ and waits for one clock cycle before latching the data. As $\overline{AS}$, $\overline{UDS}$ and $\overline{LDS}$ are negated, the shift register is cleared ready for the next cycle.

For word write operations to the Z80, the 74LS373 latch takes no part, disabled as $\overline{WREN}$ is held high by R/$\overline{W}$. The rest of the operation is substantially as for the read cycle, the 74LS245 buffers being driven in the reverse direction.

As the shared memory is word organized, advantage can be taken of the $\overline{UDS}$ and $\overline{LDS}$ lines on the interface board which are valid whether generated by the M6800 or the Z80. Once the arbitration circuitry has granted control of the interface to a processor, the presence of the shared memory request signals and $\overline{UDS}$ and $\overline{LDS}$ can be used to generate a chip enable signal for the two banks of static random access memory devices. The R/$\overline{W}$ line is also available for use as a R/$\overline{W}$ line to the memory, but in the form provided by the board it is a board direction signal, rather than a data direction signal. To overcome this, the R/$\overline{W}$ signal is exclusively ORed with the M68000 shared memory ownership signal $\overline{B}$, thus causing the signal to be inverted if the Z80 is performing the access.

8.6 The Intel 8086

The Intel 8086 was the first sixteen bit processor released by Intel and is available in a forty pin pack. Operating on sixteen bit data and using a sixteen bit wide data bus, a multiplexed address data bus is necessary to allow a twenty line address bus. In similar fashion to the M68000, the memory is organized as two independently selectable eight bit banks. The method used to achieve a sixteen bit access is different to that employed by the M68000 in that the lower byte is selected by the address bus and the higher byte is explicitly selected by the bus high enable ($\overline{BHE}$) signal. The processor can be configured in one of two modes by applying zero or five volts to the MN/MAX

pin. The present implementation uses the minimum form and thus the configuration input is tied high.

Unlike the Motorola M68000, the memory of the Intel 8086 is not organized as one continuous segment, rather all memory references are made with respect to one of four offset addresses provided by the code, data, stack and extra segment registers. A sixteen bit offset, provided by the instruction code or an addressing register is added to the value presented by the segment register. Two sections of memory are required to enable a minimum system to run. Upon reset, the processor starts to execute the instruction at location $FFFF0_{16}$, which will usually be a jump to the start of the intitiallsation program.

Interrupts to the processor are vectored, and the first kilobyte of memory is reserved for two hundred and fifty six vectors, each composed of a sixteen bit value to be loaded into the code segment register and a sixteen bit offset to be loaded into the program counter. A minimal system must therefore have memory both at the top and bottom of the address map.

Input/output devices have a separate address space selected by the M/$\overline{IO}$ pin and can be addressed with either an eight (I/O locations 0 to FF_{16}) or sixteen (locations 0 to $FFFF_{16}$) bit address. Therefore, the most frequently used devices should be placed at the bottom of the Input/output address space to allow the shorter addressing form to be used.

The requirement for memory devices to appear at both the top and bottom of the address map makes the Intel 8086 one of the most complicated to provide address decoding for, particularly on a system with a design aim of apparent board simplicity.

The Intel 8086 uses a conventional bus timing sequence by which the processor establishes the address and generates an address latch signal (ALE) and then, if writing, presents data. The processor assumes that a data

transfer can be completed in two clock periods, and any input/output or memory devices that cannot meet this timing must take the processor READY line low to request that the processor inserts wait states into the cycle. The processor can be waited indefinitely if required without difficulty as the bus cycle is not synchronized to the system clock.

The system clock is generated by an Intel 8284 clock generator device. This is designed specifically to support the 8086 processor and provides a clock signal that is one third of the frequency of the crystal used to generate the stable frequency reference. In addition, the READY line from the system is correctly synchronized with the system clock before being presented to the processor.

All memory and input/output references cause one wait state to be inserted by the on board circuitry. This is based around a 74LS107A dual JK flip flop. The 8086 only examines the state of the READY line during active bus cycles and, until $\overline{RD}$, $\overline{WR}$ or $\overline{INTA}$ (Interrupt acknowledge) become active, the flip flop is held clear causing the 8284 RDY input to be held low. Under normal cicumstances, the J input of the flip flop is tied to five volts and at the first READY sampling point of a bus cycle, the output of the flip flop will still be zero, thus causing a wait state insertion. By the next sampling point, RDY will have become high as the high input on the flip flop input will have been clocked through.

To provide a single cycle control, the input to the flip flop can be taken from a second JK flip flop also cleared by the absence of an active bus cycle. This device is in turn clocked by the output of a single cycle switch debounce circuit. The circuit is reset at the end of the bus cycle, thus causing the next cycle to be waited until the switch is operated again.

To enable the correct generation of a reset pulse from a resistor/capacitor circuit, the 8284 contains a schmitt trigger input circuit

which is connected to an open collector line running along the system backplane. On board, power on and reset push button signals are fed onto the line via open collector buffers. The resultant reset signal produced by the 8284 is passed to the processor, all on board input/output devices and an inverter, used to drive an LED, that provides a visual indication that the processor is held in the reset state.

As the Intel 8086 utilizes a multiplexed bus, provision must be made for demultiplexing the address and data information for the on board memory and input/output devices. Whereas for processors with non-multiplexed buses, the address and data lines are buffered with 74LS244 and 74LS245 devices respectively, here the multiplexed lines feed both 74LS373 latches, whose outputs form the address bus, and 74LS245 bus transceivers, connected to the data bus. In this way the processor lines are both buffered and demultiplexed. The control signals for both the address latches and data buffers are provided directly by the processor in minimum mode and, while the stated design aim of this system is to provide a backplane bus which is equivalent to that produced by the processor (in this case multiplexed), the designated peripherals suggested by the manufacturer are those from the 8080 family (using a non-multiplexed bus) rather than those from the 8085 family (a multiplexed bus). In summary, the control signals provided by the processor for demultiplexing of the bus make no allowance for the transmission of multiplexed address/data information down either the data or address buses. It is therefore argued that the 8086 should be correctly considered as a non-multiplexed bus processor composed of a processor and several support devices (in this case, latches and bus drivers).

In the minimum mode, the processor supplies the address latch enable (ALE) signal which can be used to drive the 74LS373 enable input directly, whilst the output control of the latches can be driven by the signal used by

the processor to indicate bus availability to another bus master (HLDA). The direction control input of the 74LS245s (providing that the orientation of the drivers is correct) is supplied by the Data Transmit/Receive signal (DT/$\overline{R}$). The data buffer enable control signal (DEN) must be combined with the HLDA signal as it becomes tristate during direct memory access operations, and this combination is achieved by ORing the two signals before transmission to the enable input of the 74LS245s.

The buffers provided to isolate the board from the bus are all bidirectional, thus allowing the supervisory processor to access on board memory and input/output devices. As the buses are demultiplexed, 74LS245 devices are used. Three buffers drive the twenty address lines and the four control signals that are bidirectional, M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$, and BHE. The buffers are permanently enabled and the directional control is provided by the HLDA signal. The data bus buffers operate as a single sixteen bit buffer, no attempt being made to drive only that half of the bus selected by A0 and BHE. The directional control for these buffers is provided by the HLDA and $\overline{RD}$ signals which are exclusively ORed together. The buffers are enabled whenever either the $\overline{RD}$ or $\overline{WR}$ lines are low and the off board signal (OFBD) is present (8086 controlling the bus) or absent, i.e. on board device selected (supervisory processor controlling the bus). This is achieved by exclusively ORing HLDA with the OFBD signal.

The control signals that are unidirectional are buffered by a 74LS04 inverter device. Thus $\overline{CLK}$ and $\overline{HLDA}$ are sent through the system, whilst $\overline{NMI}$, TEST and $\overline{HOLD}$ are accepted as open collector signals, inverted to the form required by the 8086 and passed to the processor. In addition, one gate of the 74LS04 drives a LED indicating that the processor has released the bus to an external device.

The board is provided with three kilowords of read only memory (six 2708

EPROMs) and two kilowords of random access memory (eight 2114 devices). To accommodate the requirement that the device (RAM or ROM) placed at the reset location is also available for the interrupt vectors, a 12L6 PAL is used to generate the address decode.

The PAL takes as inputs the demultiplexed version of A1 to A19, the M/$\overline{IO}$ signal and the outputs of two switches. The switches allow the selection of one of the four address maps as shown in Figure 8.12. Figure 8.13 shows the overall memory and input/output maps as seen by the 8086, whilst Table 6 gives the logic equations implemented in the PAL. The PAL produces the OFFBOARD signal that is used to control the bus drivers, an input/output selection signal and two signals for read only memory selection and two for random access memory selection. The output signals (apart from the OFFBOARD signal) are used to control the actions of 74LS155 demultiplexers. For the ROM and RAM devices, the two selection signals feed the select inputs of their respective 74LS155, whilst the data inputs of the 155s are fed from other control signals. For the RAM selector, A0 and $\overline{BHE}$ are used as the data to be fed to the inverting outputs of the two halves of the demultiplexer. Thus the enable inputs of a RAM device will only be taken low when the appropriate control signal (A0 or BHE) is in the correct state and either $\overline{RD}$ or $\overline{WR}$ are active, these being used to enable the 74LS155. The 74LS155 controlling the read only memory devices is similarly connected, being enabled by $\overline{RD}$ alone, thereby removing the possibility of a bus conflict if software attempts to write to an EPROM.

The input/output devices provided on the board are an 8251A universal synchronous/asynchronous receiver/transmitter, an 8255A programmable peripheral interface, an 8259A programmable interrupt controller and a write only port driving eight LEDs. These devices are connected to the lower byte of the data bus and provide the serial and parallel input/output facilities

required by the board. The 8251A has a baud rate generator that divides the processor clock to provide a range of standard transmission rates from 75 baud to 4800 baud. This circuit is based upon a 74LS393 dual four bit counter, one of whose outputs can be fed to the 8251A baud rate input by switch selection.

The programmable interrupt controller provides eight inputs that can be individually vectored by the provision of a byte during interrupt acknowledge cycles from the processor. The byte provided is interpreted as a vector number. Of these eight inputs, four (levels four to seven) may be used by the 8251A and 8255A devices since the interrupt lines of both these devices are passed through open collector drivers and onto the programmable interrupt controller via switches. If the devices are not enabled for interrupts, or the switches are not closed, these interrupt lines may be used by other devices without interference. Interrupt lines zero to three are driven by the interface to signal various conditions and the action of these inputs will be described in the discussion of the interface.

As neither $\overline{RD}$ or $\overline{WR}$ are active during interrupt acknowledge cycles, the off board buffers are not enabled so there will be no conflict as the programmable interrupt controller places the vector number on the data bus.

8.7 The Intel 8086 Interface

The Z80/Intel 8086 interface must provide the same bus width translation facilities as the Z80/M68000 interface. As the bus fed to the interface is demultiplexed, there are no additional considerations relating to the multiplexed bus provided by the processor.

As the 8086 has an input/output space, three signals from the Z80 memory management unit are used to select between the shared memory ($\overline{SMR}$), target memory ($\overline{OMS}$) and target input/output space ($\overline{TIOS}$).

The two signals required from the 8086 to indicate access to shared memory ($\overline{SMRQ}$) and the Z80 ($\overline{ZRQ}$) are produced by simple combination of the

demultiplexed address lines, qualified with the 8086 $\overline{RD}$ and $\overline{WR}$ signals.

As the interface board treats a Z80 access to the 8086 in exactly the same fashion as for a memory access (the only difference being in the state of the 8086 M/$\overline{IO}$ signal) there are thus four requests that must be examined by the on board arbitration circuitry. The wait state generation circuits are triggered by the presence of an unsatisfied request.

The arbitration function is performed by a 10L8 PAL. When a facility is requested by the Z80 which is not available, the PAL causes the $\overline{WAIT}$ line to become active. This signal is fed through an open collector buffer onto the Z80 $\overline{WAIT}$ line and the Z80 will wait until the facility is available. Once the shared memory grant signal ($\overline{SMG}$) or the other micro grant signal ($\overline{OMG}$) become active, the wait state generator circuit is released from the cleared state. The wait state generator consists of four D type flip-flops, all clocked by the Z80 clock, ϕ. The $\overline{WT}$ signal is used to insert three wait states into the Z80 bus cycle after the requested memory becomes available and the flip-flops continue to clock a high level although, at this point, only the removal of the memory request can clear the circuit. Note that the $\overline{WT}$ signal is normally active and is further modified by the arbitration PAL before being passed to the Z80. The second signal is $\overline{MW}$ which is used to place a low pulse on the 8086 $\overline{WR}$ signal during Z80 write cycles to the 8086 memory and input/output devices.

The 8086 wait signal (RDY) is also produced by the arbitration PAL. As the requested memory becomes available a state machine is used to sequence the 8086 access onto the Z80 bus. The sequencer generates the 8086 equivalent of $\overline{WT}$ discussed above as part of its function.

The $\overline{ZRQ}$ and $\overline{SMRQ}$ signals generated by the address decoder attached to the 8086 address lines are conditioned by the $\overline{HLDA}$ signal so that they cannot be generated when the Z80 owns the 8086 bus. If such an attempt is made, the

signals are diverted, and each illuminate a LED which is cleared by a switch or software in the Z80. The $\overline{ZRQ}$ is further conditioned and, if the Z80 is not available, (either due to the on board disable switch electrically isolating the Z80 or as a result of Z80 software disabling accesses), the cycle is discontinued and an interrupt is generated being passed to the 8086 PIC, illuminating an LED and setting the illegal access attempt input on the Z80 PIO. If the signal is not so intercepted, it is used to remove the set input on an SR latch, the R input of which is arranged so that when the Z80 is in control of its bus, a bus request will be generated. The purpose of this latch is to guarantee that when the 8086 finishes an access, no further access attempts will be made until Bus Acknowledge from the Z80 has been received. In this way, the Z80 executes at least one memory access cycle between consecutive 8086 direct memory access cycles (thus refreshing the board 2 dynamic random access memory) and the possibility that the 8086 might issue a new request for DMA whilst the Z80 is in the process of removing the bus acknowledge for the primary cycle, thus causing an aborted access, is also removed. The arbitration circuit monitors the Z80 $\overline{BUSAK}$ line and when it becomes active, Z80 grant ($\overline{ZGT}$) is asserted. This in turn releases the 8086 state machine.

Access requests by the Z80 are passed onto the 8086 $\overline{HOLD}$ input with little modification. The arbitration PAL is used to implement an SR flip-flop to perform the same function as that described for the Z80 bus request line. In turn, the 8086 HLDA line indicates the availability of the 8086 bus.

The grant outputs of the arbitration PAL ($\overline{SMG}$, $\overline{SMGT}$, $\overline{SMG}$, $\overline{SMGT}$, OMG) are gated to produce the address and control buffer enabling and direction signals. When the Z80 is granted either the shared memory or the 8086 bus the Z80 to shared memory control buffers are enabled, whilst the buffers are enabled in the reverse direction whenever the Z80 has acknowledged a bus

request. Similarly, the 8086 control buffers are enabled whenever the 8086 is granted the shared memory or the Z80 bus, whilst the buffers are enabled in the reverse direction by the $\overline{OMG}$ signal from the arbitration circuit. The address buffers are similarly treated except they are bidirectional and will only drive onto their respective processor buses when that device has issued a bus grant ($\overline{ZGT}$ or $\overline{OMG}$). The conditioned bus grant signals are used to remove the possibility that the buffers might drive onto the processor bus during the undefined response period (when a new bus request occurs in the presence of a previous bus grant). The data bus buffers are controlled by a second 10L8 PAL that accepts the grant signals of the arbitration PAL, A0, $\overline{BHE}$, $\overline{WR}$ and $\overline{RD}$ (these signals being produced on card by both processors, $\overline{BHE}$ under Z80 control being $\overline{A0}$). This allows both direction and enabling signals for the buffers to be produced by simple combination. Whilst this device also produces the shared memory chip enable signals, it is not responsible for the generation of the latching signal used in the translation from sixteen to eight bit bus accesses.

All 8086 accesses to either the shared memory or the Z80 are controlled by an eight bit latch and thirty two by eight bipolar programmable read only memory configured as a state machine offering nineteen states. Until the 8086 is granted either the shared memory or the Z80, the outputs of the 74LS273 latch are held clear. This in turn forces the data bus latch to be transparent should it be required by Z80 accesses to the shared memory or the 8086. Once the clear input is inactive, the 8086 clocks the programmable read only memory outputs through the latch in a sequence, dependent on the state of the single input to the system which is low only when the 8086 is performing a sixteen bit access of the Z80. The resultant waveforms generated for a word direct memory access into the Z80 are shown in Figure 8. 14. The signals produced by the state machine are the A0 bit passed to the Z80, the data bus latch control

signal and a combined $\overline{MRQ}$ and $\overline{W}$ for passing to the Z80, $\overline{W}$ being gated with the 8086 $\overline{WR}$ signal so that it is only issued during a write to the Z80.

The final output of the system is the 8086 wait signal which is used to hold the 8086 during the time required to perform two bus accesses to the Z80.

The on board PIO used to control the state of the 8086 uses a separate buffered data bus so that the PIO can monitor the Z80 instruction stream for a "RETI" instruction. The PIO generates as outputs the Z80 available signal ($\overline{ZAV}$) the Halt 8086 signal ($\overline{HOLD}$) and the 8086 synchronizing signal (TEST). Accepted as latched inputs are the illegal access attempt signals, the illegal dynamic memory access (Z80 attempting to access 8086 address corresponding to shared memory or Z80) and the interrupt lines IR1 and IR4 to IR7. The 8086 $\overline{NMI}$, $\overline{RES}$ and $\overline{HLDA}$ lines are available to the PIO as direct inputs.

To generate interrupts to the 8086, two output lines from the PIO are fed to a 74LS156 decoder. Whenever the PIO data port used to alter these selection lines is written to, a counter circuit enables the multiplexer for 500 nanoseconds. Thus a pulse is applied to the line selected by the PIO from amongst IRO to IR3, NMI, RES and the latch clear line $\overline{CLR}$, responsible for reseting all on board latches.

8.8 The Ferranti F100L

It has been noted in Chapter 4 that the Ferranti F100 is a sixteen bit microprocessor utilizing a multiplexed address and data bus. The bus timings of this processor are extremely complex, relying on translations occurring on four control lines to indicate the various stages of a bus cycle. The F100L is also unusual in that the support devices available for use with the processor include parts designed to allow the implementation of bus to bus interfaces between multiple F100 processors and to allow the easy implementation of shared resources. The bus passed along the backplane of the F100 system produced for use with the supervising processor differs from that advocated by

Ferranti in two respects. The address/data bus is tristate rather than open collector and it is not inverted. These modifications do not affect the ability of the F100 support devices to correctly operate with the bus and to simplify the interface, the controlling element of the F100 interface set, the F111L, is used on the interface to generate both the memory request signals for the shared memory and the Z80 and the correct sequence of control signals on the F100 bus when the Z80 accesses the F100L. A further difficulty presented in the design of an F100/Z80 interface (and shared with the Texas 9900) is that all access to memory and input/output devices on the F100 bus are of word length so that the bus does not support the signals necessary to allow the accessing. This is resolved in the case of the F100 by providing circuitry on the interface card that generates a 'read sixteen bits, modify eight bits, write sixteen bits' cycle whenever the Z80 performs an access into the F100.

To allow the use of support boards which cannot cope with a multiplexed bus, the F100 system also offers a demultiplexed card which allows the use of standard cards such as the modified "Softy". This card also contains an F111 interface control device which again allows the production of bus with the more usual type of control signals whilst being considerably simpler than the circuit of the interface card as the support cards do not perform direct memory access into the F100 memory.

8.9 The Zilog Z8000

The Z8000 implementation is based around the more complex of the family microprocessors, the Z8001 which utilizes a multiplexed address and data bus, transmitted down the backplane in multiplexed form. The data and address paths provided by the interface board follow the general pattern of those associated with the F100 without the additional complexity introduced by the need for word only accesses or stringent bus timings. As the Z8001 employs a

multiplexed bus it is possible to allow all the address and segment lines (effectively all address lines) to pass to the interface board.

These lines total twenty three, more than the number allowed for by the Z80 memory management unit. It is therefore necessary to extend the memory management unit on the interface board to provide additional lines into the Z8001 system.

This is done by duplication of the part of the memory management unit circuitry of board 2, but, as the other facilities such as master/slave attributes and slave access address are not required, the circuitry is considerably simpler. As a multiplexed bus is employed, it is also necessary to offer a demultiplexer card so that other boards can be employed. Once again, because of the relative simplicity of the Z8001 bus timings, the board is simpler than that required for the F100, requiring some seven 74LS series integrated circuits to provide a suitably demultiplexed bus.

A problem associated with the demultiplexer boards is that the data bus buffers on the board must only enable towards the processor when data is to be read through the interface board as other boards on the multiplexed bus might be trying to drive the bus at other times. This requires the equivalent of an 'on board' signal to be provided by any board plugged into the system through the multiplexer. This is achieved by providing an open collector line to signal to the demultiplexer that action is required.

8.10 Summary

The microcomputer boards described all provide a minimal implimentation of their particular microprocessor. In each case, the processor has been provided with enough memory and input/output devices to allow the board to be used for demonstration or experimentation. The complexity of the boards has been kept to a minimum without sacrificing the expandability required to allow the boards to be used for prototype work.

The implementation of the interface scheme described in Chapter 7 has been shown for several processors. Using the interface, the supervisory processor can access any memory or input/output location available to the slave microprocessor. In addition, the slave can, if allowed by the supervisory processor, access the memory of the supervisor for the placing and retrieval of data or instructions, using any addresing mode. Either processor can access the shared memory on a first come, first served basis, only being delayed when both processors attempt to access the memory simultaneously. The supervisory processor can halt, reset and interrupt the slave through the input/output device on the interface card. In the particular case when the reset vector locations of the slave are occupied by random access memory the supervisor has total control of slave execution. Should the user place EPROM at the reset vector locations, the slave system can operate independently and can be used in prototype systems. As the supervisor can access the memory of the slave, it is possible to examine the stack and, by means of interrupts, single step the siave. The input/output of the slave is aiso available and the user can affect any devices connected to the input/output ports, either from monitor commands, or from programs running in the supervisor. When the slave accesses the memory of the supervisor, the memory manager provides address translation, giving control over those sections of master memory that can be altered by the slave.

The Softy board provides the system with both EPROM programming and data display facilities. As the Softy can be placed into the memory map of either processor, its ability to show stack operations can be of benefit to those developing software on the supervisor, as well as those examining a new slave.

Although only a relatively small number of target systems have been described in any detail, those chosen show sufficient diversity to confirm that the concept of a supervisory/slave microprocessor system is capable of

operation with a wide range of both eight and sixteen bit microprocessors exhibiting widely different bus structures. Further details of circuitry can be found in the complete circuit diagrams for several of the boards which are provided in Appendix 1.

9 Support Peripherals

Whilst the system specified offers many useful features to the user, various facilities are required to enable full use of the processing power provided by the microprocessor. Adequate provision must be made for the generation, display and storage of information in various forms. This is achieved by the use of a set of peripheral cards which provide all the facilities for a specific application, except for the processor and large quantities of memory, both of which can be supplied by the Z80 system. This Chapter will describe one such device (a floppy disk interface) in detail and then discuss other devices available.

9.1 Floppy Disk Interface

Floppy disks are a common medium for storing programs and data for use with microprocessor systems. As they allow random access to the disk surface, they are more convenient than tape storage and, whilst slow and with little storage, they offer low media costs. The major characteristics of a floppy disk that influence the design of an interface to a microprocessor are the data rate and the method used to encode that data for storage on that disk, whilst less significant characteristics include the method used to control the head movement over the surface. The high degree of compatibility between the products of various disk drive manufacturers has made the provision of LSI interface devices to ease the design task an economic proposal and a member of the Western Digital 179X family, the 1793 is used in this investigation.

Whilst presenting a bus oriented signal set to the processor, it generates and accepts the signals presented by the disk drive and offers full control over all such signals. Two areas in which additional circuitry is required are the provision of an interface to floppy disks providing double density storage using modified frequency modulation (MFM) and the movement of data from the WD1793 to the host at rates of one byte every eight microseconds

(worst case). The method of providing support for double density disk storage is derived from Western Digital literature and is based upon Western Digital support chips.

Two methods may be employed to allow data transfers to a Z80 at the rate of one byte every eight microseconds. The more commonly adopted solution is the provision of a direct memory access controller which, when signalled by the peripheral device, places an address on the bus corresponding to the required memory location and enables the input/output device thereby allowing the data to be transferred with no software overhead. A second approach is to place the Z80 into a small program loop in which it accesses the peripheral location responsible for providing the data. If the data is not available, the Z80 is waited until the data becomes available, where the cycle ends, the data is stored and the Z80 re-accesses the peripheral data port.

Both the above solutions are unsatisfactory in this case. The provision of direct memory access facilities is not complex when only the processor and the direct memory access controller have the ability to take control of the bus and arbitration is easily achieved, especially if the direct memory access device used is selected from the manufacturers support devices. In the Z80 system described, the presence of the memory management unit, and the ability of the slave to request a direct memory access transfer through the interface board would increase the complexity of the bus arbitration circuitry and bus timings.

The second proposed method is also unacceptable as it involves the creation of extremely long periods when the bus is unavailable, (up to one complete rotation period of the disk). This has two effects. Firstly, the Z80 is not able to refresh the board 2 dynamic random access memory and, secondly, if a slave processor requires access to the Z80 it will also enter a long wait state until the bus access has finished. If the Z80 is now removed from the

bus by the slave direct memory access request, data from the disk might be lost, requiring that the transfer be repeated.

To overcome the objections outlined above, a board has been designed that contains the Western Digital floppy disk controller and support devices, one kilobyte of random access memory and a simple direct memory access controller. The Z80 now acts as a direct memory accessing device with bus arbitration between the Z80 and the on board direct memory access controller being on the basis of first come first served. Once the transfer by the WD device is complete, the Z80 moves data from the on board random access memory to the designated buffer location in the main memory of the system.

The described technique is inefficient compared to the full direct memory access facilities but does offer a number of advantages in that the resultant board can be used with little change on any system, with or without direct memory access whilst data may also be abstracted from the buffer rather than all being transferred to main memory.

The main elements of the floppy disk interface are standard and they will not be discussed. Further information can be found in [18].

Several modifications are included, primarily the clock frequency is variable to allow the use of either five and a quarter inch or eight inch disks, whilst the WD1793 input signal DDEN(double density enable) is also controllable.

The board selection circuitry recognises two areas. The first contains the floppy disk controller, disk parameter control port and direct memory access control port which are decoded as input/output devices with addresses between $F8_{16}$ and FF_{16}. The second area contains the on board memory buffer which is decoded to lie between address $FFC00_{16}$ and $FFFFF_{16}$ (use is made of the Z80 memory management unit to supply the top four lines). As the high order address lines are not required for the direct memory access scheme,

their decoding is by simple combination. The connection of addresses to the one kilobyte buffer is made via 74LS157 multiplexers and the processor address lines are permanently enabled onto the inputs of the 74LS157s' thereby allowing these lines to be used to provide the decode for the input/output block. The R/$\overline{W}$ address and chip select lines to the floppy disk controller are also fed via a 74LS157 multiplexer. Under non-direct memory access conditions, these lines are those produced by the Z80 and the input/output decode circuit. During a direct memory access cycle, all 74LS157 devices are switched so that the memory buffer addresses are those provided by the direct memory access controller, whilst those fed to the floppy disk controller are the signals necessary to select the data port and a R/$\overline{W}$ signal derived from the direct memory access control latch, the inverse of which is fed to the buffer memory during this time. The net effect is that both the floppy disk controller and the buffer are enabled with one reading and one writing, the address for the buffer is provided by the direct memory access controller and a data byte is transferred. Should the floppy disk controller request a direct memory transfer as the processor is accessing the floppy disk controller (to read a status register), both the Z80 and the direct memory access controller will require access to the buffer and floppy disk controller. This is resolved by an arbitration circuit which samples each of the two request lines in turn, granting the access to the first active request found. The circuit is based upon a 74LS139 being driven by a counter, in turn operated from the sixteen megahertz on board clock. The clock signal to the counter is gated so that the count may be held at any point. The counter and 74LS139 produce two non-overlapping clock pulses with a mark/space ratio of 1:3. This leaves a gap equivalent to the mark duration between each request, allowing the device selected (direct memory access controller or Z80) time to hold the clock signal to the counter.

As the floppy disk controller will leave a direct memory access request pending until satisfied, there is no requirement for a wait state generator except for that controlling Z80 accesses.

As soon as the Z80 requests either the buffer memory or the floppy disk controller, a wait signal is generated. This is maintained until the Z80 is granted access, at which point a monostable is triggered to generate a further four hundred and fifty nanoseconds of wait state, thus ensuring that adequate time is allowed for random access memory and floppy disk controller accesses. At the end of an access, the request generated by the on board decode is removed, thus releasing the arbitration circuitry. Once a direct memory access has been granted, all 74LS157 devices select the direct memory access values for their outputs and two monostables are triggered, one controlling the length of the access, the other the point at which the direct memory access counters are incremented (*after* a DMA cycle).

The direct memory access control is simplistic, the value of the counts can only be cleared and the buffer starts at on board random access memory address 0. The direction of the direct memory access transfer and the direct memory access enable signal are controlled by a latch, selected at input/output address FE_{16}. In this latch, the top bit is used to enable the direct memory access controller when it is set, whilst bit five is used to select the direction of the transfer. Positioning of these two bits is such that if the read sector or write sector commands are also written to the direct memory access controller, the controller will be correctly set for the coming transfer, thus saving a separate accumulator load instruction and making the system easier to control.

The floppy disk interface board has been used to successfully implement the CP/M operating system on the Z80 microcomputer described. The implementation makes use of the memory manager to remove the erasable

programmable read only only memories, containing the monitor, that are located at zero (as the system random access memory must be contiguous from zero upwards) and to bring into the map both monitor EPROMs, floppy disk support EPROMs and floppy disk controller buffer as necessary, whilst utilizing a sixty four kilobyte map when programs are running. The ability to use the EPROMs in this way reduces the amount of software that must be present in the memory map whilst the users program is running, giving increased workspace to such programs.

9.2 Prestel Interface

The British Telecom Prestel system is a remote access computer allowing the retrieval of general information. Access is via a Prestel terminal that allows the user to communicate with the remote computer at a rate of seventy five baud, whilst the computer/terminal link operates at twelve hundred baud. The connection to the remote computer is established via the public switched telephone network (PSTN) and the terminal must have autodial capability and an internal modem whilst meeting the stringent safety specifications of British Telecom. Once such a terminal is available, it can also access the X25 Packet Switching Service, allowing connection to a wide range of computer systems similarly equipped. An interface card has been produced that provides the correct format of screen (similar to that of the BBC Ceefax and ITV Oracle systems) and utilizes the Mullard LUCY device to provide autodialling pulses, communication protocols etcetera.

A Mullard line interface board providing voltage isolation and filtering was used to ensure the device met safety requirements. The Z80 computer supplies the processor, the program storage (EPROM is used), a large quantity of random access memory for print buffers, keyboard and serial ports for connecting to a printer or remote terminal. The software provided for the system is comprehensive, offering many features not normally available on such

terminals. The user may request a copy of the contents of the screen, and, as the printer in use has one hundred and twenty eight columns, whilst the Prestel screen is only forty, three pages will be buffered so that all three can be printed at once, thus reducing paper usage. Alternatively, pages may be printed automatically, a copy being produced of every page transmitted. The call to remote computers requires that the user provides a password, held in encoded form in the EPROM, as the terminal uses normal university lines for access. Key strokes are internally buffered so that the seventy five baud transmission rate does not cause occasional characters to be lost. Should the user request that a page on the screen be printed whilst the print buffer is full and printing, then transmission to the remote computer is halted until the buffer becomes available, when the keyboard buffer is sent. At any time the user may force the print buffer to be printed, even though three pages have not been collected so that the user may refer to items in the buffer but not on the screen. The keyboard buffer auto locking mechanism can lead to the user not being able to communicate with the remote computer in the event of, for example, the printer failing. Therefore, an escape sequence override is available to either purge the print buffer or to force the transmission regardless of the state of the buffer.

The terminal has been used extremely successfully to access the Prestel system and the X25 Packet Switching Service.

9.3 High Resolution Graphics

The basic visual display unit system available to the supervising processor has very limited graphics capability. As many applications require high resolution graphics, a peripheral card has been developed which offers five hundred and twelve by five hundred and twelve pixels with an overlaid cross-hair cursor and touch operated cursor movement control whilst allowing the processor to read the correct value of the cursor at any time. The

graphics display board appears to the Z80 as 262,144 memory locations, each of one bit. These locations are accessed using the memory manager and software can set or clear a bit at each location. In addition, the entire screen can be set or cleared with a single command.

The design is based upon 4116 dynamic random access memory devices, no refresh circuitry being required as the constant access by the screen display circuitry provides this. Basic screen timing is provided by a Ferranti ZNA134 television synchronizing pulse generator integrated circuit. This provides highly accurate line and frame blanking and sync pulses, derived from a 10.25 megahertz clock. These outputs are used to clock line counters whilst the 10.25 megahertz signal is used to provide the dot clock with the counters being cleared by the line sync pulses. As selected by the video circuitry, the dynamic RAM devices are organised as two banks of eight bits each, one for the odd field and one for the even. This is to allow the random access memory time to produce the data at a reasonable rate, rather than the 10.25 megahertz required by the dot rate. Eight bits of screen information are loaded into a shift register simultaneously, thus reducing the random access memory rate by a factor of eight. When accessed by the Z80, the bottom three address lines (A0 to A2) select which of the eight memory devices in the bank is to be enabled. The bottom nine address lines correspond to the X co-ordinate of the screen, whilst the next nine lines provide the Y co-ordinate.

The cursor facility operates by generating an output every time the row or column counters match the current value of the cursor counter. When this occurs, the output is exclusively ORed with the screen generated data and appears as white on black or black on white. The values in the cursor X and Y registers are affected by the cursor control buttons, these causing their respective counter to increment or decrement. As the outputs of these counters are available to the processor it can identify the current location of the

cursor. Note that the cursor information is not held in the screen bit plane and thus does not require complex cursor management software to calculate whenever a bit should be illuminated to provide a cursor. The cursor is currently full screen as a match on either the X or the Y is sufficient to generate a line. This can be altered to generate a smaller cross-hair if necessary by combination of the output from the two counters.

9.4 Audio Spectrum Analyser

A low frequency spectrum analyser has been produced that utilizes the high resolution graphics board described above to allow the display of a spectrum acquired in the range of twenty hertz to twenty kilohertz with a resolution of one hundred hertz.

Several possible methods exist for obtaining the spectrum of a waveform. The waveform can be fed to a large number of fixed frequency bandpass filters or one bandpass filter with a movable centre frequency. However, the first of these methods is highly expensive whilst the second is difficult to implement satisfactorily. The usual technique mixes the incoming waveform with a variable frequency generated by the analyser. Once modulated, the input waveform will have had its spectrum moved by the amount of the modulation frequency. If this is presented to a normal bandpass filter, the amplitude of that frequency component can now be measured. If the modulation frequency is changed, the portion of the input spectrum now passed by the filter is altered. In this way, the entire input spectrum can be examined with one filter. Instruments using this technique are known as swept spectrum frequency analyzers. The filter used has a bandwidth of one hundred hertz and it is therefore necessary to be able to place the desired part of the spectrum at the centre frequency of the filter to obtain an accurate reading. The modulation frequency is generated by a voltage controlled oscillator, (a 74LS124) whose input voltage is produced by a digital to analogue converter.

To allow accurate placement of the input frequency the digital to analogue converter must produce a sufficient number of discrete frequencies from the voltage control oscillator. Over a frequency band of twenty hertz to twenty kilohertz, a one hundred hertz filter makes available two hundred distinct measurements and the digital to analogue converter must therefore allow considerably better resolution than this to achieve accuracy. A ten bit digital to analogue converter is therefore used which allows the modulation frequency to be placed at any multiple of 19.5 hertz, thus assuring that the centre of the input waveform spectrum of a given frequency is not more than ten hertz removed from the centre of the bandpass filter.

The modulated waveform is rectified and an envelope produced, measured by an eight bit analogue to digital converter (RS427). The resultant value is available to the Z80 for processing. Approximately four seconds is required to scan the analyzer over the entire spectrum, leaving the gathered data in random access memory inside the Z80 system. This is available for further processing, storage or display, all performed off line so that the input spectrum may be swept as fast as practicable.

9.5 Other System Peripherals

Interfaces have also been produced that allow the Z80 to accept and generate analogue data at low data rates. These interfaces are based upon standard analogue to digital and digital to analogue converters and both low precision, low cost and high precision versions have been designed to allow the provision of analogue input/output for both general purpose and specialist use.

Arguably a system peripheral, use has been made of a two hundred and fifty six kilobyte RAM board (designed for use with a stand alone Motorola M68000 system utilizing a pinout compatible with that of the M68000 system described in Chapter 8) and the enlarged addressing space of the Z80 system,

to provide the CP/M operating system with a virtual disk. Such an arrangement appears as a disk that operates many times faster than either floppy or winchester disks. Whilst this arrangement is becoming more common on microcomputer systems, the memory management unit (provided to allow adequate access to slave microcomputers) makes the integration of such a facility extremely simple.

The high data rates of winchester disks makes mandatory the employment of direct memory access with little request overhead. This in turn has caused the winchester disk manufacturers to provide interface boards operating in a fashion similar to that discussed for the floppy disk interface in Section 9.1. To extend the disk storage provided by the supervisory system and allow the possibility of disk cost being shared between several systems, a commercial winchester disk controller has been interfaced to the system.

The provision of logic analysis facilities is a major feature of large development systems. The high data capture rate is usually achieved by an ECL front end storing data in high speed random access memory, which can then be examined by an NMOS microprocessor operating at normal bus speeds. To examine the provision of low cost logic analysis facilities, NMOS dynamic random access memory has been interfaced to an ECL front end that shifts the incoming data through a register, thereby allowing several low speed memory devices to appear as one high speed device. In consequence, as sixteen kilobit dynamic memories were used, the number of samples that can be taken is extremely high, giving the ability to examine waveforms both for the cause and effect of a fault, (even though they may be separated by a considerable time period) at reasonable time resolution. This is notably different from the short time period, good resolution/long time period, poor resolution offered by devices using fewer, high speed memories.

9.6 Summary

A microprocessor and its peripherals are normally designed to operate in a known configuration. This leads to the placing of a considerable data movement and examination burden on the microprocessor in an effort to reduce design complexity and cost. In a system where the microprocessor (or indeed the bus) is not necessarily available to the interface at all times, other methods must be employed to ensure the correct functioning of the system. Such techniques are employed on mainframe and minicomputers, where the use of intelligent peripheral controllers is increasing, primarily to provide greater processing through-put by reducing the input/output burden on the central processor.

A consequence of designing interfaces that appear as memory devices (i.e. not requiring direct memory access or time critical transfers) is that the bus interface becomes considerably simpler. Just as the common static memory devices can be used with a wide range of microprocessors, peripheral boards with the same characteristics as memory devices can also be used with a wide range of microprocessors.

Although all the above interfaces were designed primarily for use with the Z80, the provision of a personality socket (similar to that employed by the "Softy") would allow them to be used with the slave microprocessors, for either examination of the processor's suitability for various tasks, or to allow the up-grading of a processor should research needs change.

10. Conclusions

Examination of the equipment used for initially learning about microprocessor systems and the techniques used for designing products shows that a considerable disparity exists between the two. As the fundamental objectives of both activities are to examine and understand the behaviour of the system presented, this is all the more suprising. The reason for this disparity can be traced to the extremely high cost of equipment used for production design and testing (typically in-circuit emulators) and the inability of organisations involved in education and training to provide these facilities in quantities sufficient to provide enough working positions for education purposes.

In-circuit emulation, requiring the imitation of a microprocessor by the supervising system is both expensive and complex. The major benefit offered by in-circuit emulation is the exercising of a microcomputer, designed specifically for inclusion in an end product, with little or no necessity for incorporating circuitry in the microcomputer to enable the examination to take place.

As a consequence of the high cost of in-circuit emulators, it is generally uneconomic to use them for general purpose computing tasks, a fact recognised by the manufacturers. The response of such manufacturers has been to move software support onto host minicomputers supporting several users, whilst each in-circuit emulation station has a microcomputer responsible for downloading files from the host and controlling the in-circuit emulation process.

The system examined in this dissertation removes the ability of the in-circuit emulator to examine any microcomputer based upon a given microprocessor and replaces it with a system able to examine a representative microcomputer based on that microprocessor. As in all cases the microcomputer

may be extended, the system may be used for the examination of designs through many of the initial stages of design and production. The presence of the representative microcomputer makes the system valuable for education and familiarization as a design based upon that microprocessor is available for inspection and evaluation.

As the system is composed of several processors, (the supervisor, the slave and the INS8060 of the Softy) it can be used to demonstrate the principles of multi-processor systems. As it uses the most complex of the possible multi-processor interfacing techniques, it is possible to simulate the characteristics of other interfaces. This can be achieved by the use of pseudo input/output devices in the shared memory area with, if necessary, the provision of 'input/output' interrupts generated by the Z80 through the interface.

The hardware overhead attached to the slave microcomputer is low, being primarily the provision of a buffering schemes that allow direct memory access onto the microprocessor card itself. The overhead absorbed by the supervisory processor is limited to the provision of the memory management unit which, when present, considerably enhances the capability of the supervisory processor as a general purpose computer.

The major element of the scheme is the interface and it can be seen that the majority of the card (the data and address paths) show relatively little change for boards designed to provide interfacing facilities for microprocessors that appear markedly different. This suggests the possibility of a general purpose interface card that could, by the provision of a personality socket to allow control signal conversion, provide the data and address paths (and those control functions which are common over many interface boards) for several different microprocessor families, thus reducing the cost still further.

The system discussed is based around the Z80 as, when originally designed, the sixteen bit microprocessors where expensive and scarce. Under these circumstances, the Z80 offered (and still offers) a wide range of software (running under CP/M), the ability to refresh dynamic random access memory thereby reducing the cost of system memory and a bus structure able to support the needs of the interface scheme.

Various facilities provided by the sixteen bit microprocessors make them a more sensible choice under present circumstances. Notably the provision of bus cycle abort and re-try and the large addressing space associated with these processors simplifies the design of the system considerably. Microprocessors such as the M68000, which have single chip memory management units offering many facilities, would replace large areas of circuitry as the memory management functions were replaced by the memory management unit and twenty three bit address bus of the processor. Additionally, interfacing the microprocessor to sixteen bit slave microprocessors with word organised memory (such as the F100L and Texas 9900) would involve less difficulty as all slave bus accesses could be word organised thus removing the need for two eight bit bus cycles to be translated to one sixteen bit cycle.

The ability of the slave microcomputer to access the supervisory system was originally implemented to allow slaves with small static random access memory devices to use the larger areas automatically refreshed by the Z80. The falling cost of memory devices and the improvements in dynamic memory refresh controllers now render this ability less attractive and such needs can now be better served by the provision of high capacity self refreshing dynamic RAM boards provided with a personality socket. This further reduces the complexity and cost of the interface boards whilst improving the overall memory capacity of the system.

Appendix 2 contains the descriptive leaflet for the commercial version of

the unit described in this dissertation. This unit contains some of the improvements described above, but retains the Z80 microprocessor for the supervisory microcomputer.

This dissertation has shown that the new development system proposed fills a gap between existing microcomputer examination techniques. As the complexity of the system is less than that of apparatus offering comparable facilities, the system will be more cost effective at all levels of development and design below full hardware/software integration. The potential savings are increased if account is taken of the time saved by the provision of a working minimal system, particularly as this may be expanded and used as the basis of any prototype not subject to severe size constraints. The ability of the system to support a wide range of microprocessors allows ready comparison of different microcomputer families, their strengths and weaknesses, essential for the correct choice of a family for a particular task.

11. References

1. Noyce R.N., Hoff M.E. 'A History of Microprocessor Development at Intel' I.E.E.E. Micro February 1981

2. I.E.E.E. 'Special Issue on Microprocessors in Electrical Engineering Education' I.E.E.E. Transactions on Education, Vol. E-24 No. 1 February 1981

3. I.E.E.E. As (2) I.E.E.E. Transactions on Education, Vol. E-24 No. 2 May 1981

4. I.E.E.E. 'Microprocessors and Education' Computer, Vol. 10 No. 1 January 1977

5. 'Microprocessor Education', Euromicro Newsletter Vol. 3 No. 2 April 1977

6. Cahill S.J., Laverty S.J., McCorkell C. 'Accomodations for the microprocessor' Reference 3, pp. 173-176

7. Bray D. W., Crist S. C., Meyer R. A. 'A microcomputer Laboratory and Its Courses', Reference 3 pp. 149-154

8. Elmquist K. A., Fullmer H., Gustavson D. B., Morrow G. 'Standard specification for S-100 Bus Interface Devices', I. E. E. E. Computer pp. 28-52 July 1979

9. Gilbert N. Department of Sociology, University of Surrey private communication.

10. Lumley R. M. 'An Industrial Microcomputer Education Program' Reference 3 pp. 136-141

11. Glover J. R., Bargainer J. D. 'Integrating Hardware and Software in a Computer Engineering Laboratory', Reference 2 pp 22-27

12. Holdstock K. 'An Interface Between a PDP 11/20 and an M6800' Final Year Undergraduate Project, University of Bath, 1979.

13. Selwyn C. J. 'A Ferranti F100 as a Universal Peripheral for a PDP 11/20' Final Year Undergraduate Project, University of Bath, 1979.

14. Kelly-Bootle S. The Devil's DP Dictionary, McGraw-Hill, 1981

15. Motorola Microprocessor Products Data Book, Motorola

16. Motorola Memory Products Data Book, Motorola

17. Motorola M68000 16-Bit Microprocessor User's Manual

18. Western Digital FD179X Application Notes, November 1980

12. Bibliography

This bibliography contains the full indexes of the journals given as references two to five, with a brief description of the approach discussed by the authors.

'Special Issue on Microprocessors in Electrical Engineering Education'

I.E.E.E. Transactions on Education, Vol. E-24 No. 1, February 1981

Microprocessors in Electrical Engineering Education Survey

J.J. Jonsson pp. 3-8

The author gives the result of a survey of American universities, included are brief statistics on equipment used.

A Microprocessor Laboratory for Electrical Engineering Seniors

D.F. Hanson pp. 8-14

The system described is based upon the AIM-65 single board computer, with the addition of in house interface boards.

A Course Sequence in Microprocessor-Based Digital Systems Design

T. Mudge pp. 14-21

Z80 based stand-alone microcomputers with floppy disks, Intel 8086 single board computers and AMD 2900 bit-slice based computers.

Integrating Hardware and Software in a Computer Engineering Laboratory

J.R. Glover, J.D. Bargainer pp. 22-27

The laboratory discussed is based upon a PDP 11/70 providing software support and downloading facilities to single board computers based on the

Intel 8085.

A Systems Approach to Teaching Microcomputers

M. D. Freedman, L. B. Evans and M. H. Miller pp. 28-31

The authors propose a design language to assist students to specify and produce microcomputer software.

Goals for a Microcomputer Laboratory in a Computer Science Department

M. A. Baltrush pp. 32-34

A Z80 system with floppy disks is used to develop programs which are the downloaded into Intel 8080 based single board computers.

Microcomputers in the Engineering Curriculum

J. B. Snyder pp. 35-37

M6800 and Intel 8080 based microcomputers with floppy disks are used to introduce students to assembly language, PDP 8 and PDP 11 computers are available for comparison.

Teaching Microcomputer Programming with Application-Oriented Problems

D. B. Barlow, D. P. Agrawal pp. 38-43

Each student is provided with a Motorola MEK6800D2 evaluation board, they then move onto Z80, 8080 and M6800 based systems with floppy disk.

A Design-Oriented Microprocessor Laboratory

S. V. Iyengar, L. L. Kinney pp. 43-46

All student positions are based upon the MEK6800D1 evaluation board with interface boards providing a range of experiments.

Creating a Microcomputer Facility

J.H. Aylor, E.J. White pp. 47-50

A central time-sharing computer is used to provide support. It downloads programs into microcomputers with a standard bus structure which can accept one of six different eight bit bus processors.

"Hands-On" Microprocessor Education at the University of Rhode Island

W.J. Ohley, D.W. Tufts pp. 51-54

Teaching is based exclusively on MEK6800D2 kits with a deliberate decision to restrict student experience to one product.

Microprocessor Education in Industrial and Systems Engineering

B.L. Capehart, J.R. Dean, T.M. Kisko,

D.K. Swift, D.S. Dailor pp. 54-59

Apple microcomputers with floppy disks and 6502 based single board computers provide a range of equipment for complex or simple experiments.

A Microcomputer Laboratory Work Station

D.J. Ahlgren pp. 59-62

A time-sharing computer supports MEK6800D2 kits through serial lines. In addition, a software simulation of a microcoded machine based upon AMD 2901 devices is available.

Microprocessor-Based Automated Digital

System Laboratory Checking Station

A. Margalith, M.E. Fearon, H.W. Mergler pp. 62-67

An obsolete Intel 8008 based microcomputer is used to test logic

experiments performed by the students.

Introduction of Microcomputers to a Small Electrical Engineering Department, with Heavy Emphasis on Lab Work Using Turn-Key Microcomputers

H.R. Skutt pp. 67-68

TRS-80 level II microcomputers and BASIC are used to drive commercially produced interface boards. Assembly language is available to investigate the structure of the microprocessor.

An Interdisciplinary Course in Real-Time Computing

D.E. Thomas pp. 69-74

A PDP 11/34 running UNIX is used to download programs into LSI-11s. The course is available to all engineering students.

An Introductory Course in Microprocessors for Freshmen

R.D. Klafter pp. 74-77

AIM-65 single board microcomputers offer assembler and BASIC.

A Microprocessor System for the Discrete-Time Control Laboratory

J.W. Steadman, R.G. Jacquot, M.N. Hepworth pp. 78-86

M6800 based single board computers are used with analogue to digital and digital to analogue converters to allow students to undertake digital control experiments.

Purdue's Junior-Level Control Laboratory with Microprocessors

D.B. Gill, V.B. Haas pp. 82-86

A PDP 11/70 downloads to Intel 8080 systems connected to analogue

computers. Pre-written software modules ease the transition to microprocessor control.

Microprocessor-Aided Instruction in Power System Protection

F.C. Trutt pp. 87-89

An Apple-II with floppy disk allows students to design protection systems and assess their suitability.

Microprocessors: The Key to Cost Effective Instrumentation

D.R. Voltmer, P.T. Hulina pp. 89-92

An un-named microcomputer system provides insrumentation facilities, one unit replacing large quantities of conventional equipment.

Microprocessors in the San Diego State University Scientific Instrumentation Program

L.J. Burnett, G.L. Shackleford, J.E. Solomon, H.B. Shore pp. 93-95

Student positions can access either a PDP-8E or a single board microcomputer based upon a Z80. Several of these systems share a Z80 unit with floppy disks by means of a parallel data transfer bus. The positions contain 16 Kilobytes of RAM and can run a downloaded BASIC or assembler/debugger package.

A Lecture/Laboratory Course on Microcomputer-Based Medical Instrumentation

W.J. Tompkins, N.V. Thakor pp. 96-101

The RCA COSMAC 1802 based ELF single board microcomputer is used as the basis of a laboratory course. As part of the course, students must produce biomedical instrumentation, and test it.

'Special Issue on Microprocessors in Electrical Engineering Education'

I. E. E. E. Transactions on Education, Vol. E-24 No. 2, May 1981

A Program of Blending Hardware/Software on Microprocessors In Education

W. C. Lin pp. 108-112

PDP 11/34 supports four LSI-11s. In addition Intel 8080 microcomputers with floppy disk run CP/M and single board microcomputers based on the Texas 9900 are available for experimentation.

A Bit-Sliced Computer Design Laboratory

D. L. Dietmeyer, C. R. Kime pp. 113-119

A bit-s; ice system has a writable control store. This can be manipulated with a Z80 based single board computer, which can in turn download programs and data from a minicomputer.

A Bit-Sliced Microprocessor System for Teaching Microprogramming

C. H. Roth, G. Minassian pp. 119-122

A mainframe can download programs into an MDS-80. this can affect the contents of the writable control store associated with an AMD2901 based bit-slice microcomputer. The MDS-80/bit-slice interface is through a parallel port.

An MOS/LSI Course as an Integral Part of Microprocessor Education

J. H. Nevin pp. 123-125

The author points to the need for an understanding of the technology

responsible for the advent of the microprocessor.

MIDAS: A Microprocessor Display and Animation System

R. F. Gurwitz, R. T. Fleming, A. van Dam pp. 126-133

A graphical simulation of a microcomputer based on the Intel 8080 is discussed. The simulation shows the movement of data and addresses through the 8080, which can be reset, interrupted or single stepped by the student.

Continuing with Microprocessor Education

T. W. Davis pp. 133-136

A microprocessor up-date course for practicing engineers is dicussed. The laboratory equipment is based on S100 boards with Intel 8080 or Motorola M6800 processors.

An Industrial Microcomputer Education Program

R. M. Lumiey pp. 136-141

Bell Laboratories in house teaching unit is described. The laboratory is based on Textronix 8001/8002 development systems connected to a PDP 11/45.

Graduate Computer Science and Engineering Education for the U. S. Army at the Air Force Institute of Technology

A. A. Ross, J. E. Urban pp. 142-146

The problems of up-dating armed forces staff are discussed.

Microprocessors in Preengineering

R. A. Carlsen, J. T. Holmes pp. 146-149

Intel 8080 evaluation cards are connected to a central minicomputer via

serial lines, other microprocessor families are being added.

A Microcomputer Laboratory and Its Courses

D.W. Bray, S.C. Crist, R.A. Meyer pp. 149-154

Intel 8080 based microcomputers with floppy disks are used for interfacing experiments. The machines offer BASIC, assemblers, editors and an 8080 simulator running on the 8080.

Buying Time with Talent: The Microprocessor in the Liberal Arts College

J.G. Zornig pp. 155-159

A course using LSI11s and single board M6800s is described. Software support is provided by a DECSsystem 20/60.

Microcomputer Education in India

V.V. Athani pp. 160-162

The author examines the place of microprocessor courses in the current syllabus. Desirable laboratory equipment consists of many single board computers with hexadecimal input/output, fewer with QWERTY input/output and in-circuit emulation facilities.

A Microprocessor Integrated Laboratory

C. Conte, D. Del Corso, M. Giordana, F. Gregoretti, V. Pozzolo pp. 162-165

A PDP 11 provides software support for a range of single board microcomputers. In addition, some microcomputers have floppy disk available. The majority of units are single board, hexadecimal input/output devices.

Microprocessor Laboratory Procedure to Introduce Digital Filtering

A. G. Bolton pp. 165-168

A PACE microprocessor development is used as a digital filter.

An Undergraduate Laboratory for Digital Control and Signal Processing

D. P. Petersen pp. 168-173

Single board computers based on M6800 and Z80 microprocessors are used as the basis of digital controllers.

Accomodations for the Microprocessor

S. J. Cahill, S. J. Laverty, C. McCorkell pp. 173-176

M6800 boards are used to produce student workstations. Each has sufficient ROM to hold required support tools. In addition, students can build simple microcomputers based on the M6802 on bread-boards.

Microprocessors: Problems of Getting Started

O. R. Omotayo pp. 176-177

The author discusses the difficulties faced by mature students and approaches to overcome such difficulties are given.

An Introductory Microprocessor Course

F. E. Holmes pp. 178-180

Z80 based S100 single board computers are used for an optional course, organized by the local I. E. E. E. student branch.

Teaching Microprocessors to Exceptional Students

H. Brey pp. 180-181

S100 systems running the CP/M operating system allow interested students

to experiment with microcomputers both outside, and inside the formal education process.

Self-Paced and Error-Free Code Microprocessor Courses

H. B. Demuth, A. C. Demuth, D. R. Ballard pp. 181-183

A course, using Intel 8080 and 8086 microprocessors with both assembler and PL/M, is discussed. Each student must design a board based on the Intel 8086. Support is based on an Intel MDS-230.

Microprocessor-Just Another Circuit Element

J. L. Pokoski pp. 183-185

The 6502 based KIM-1 is used to provide initial access to microcomputer systems. Later courses use the Intel 8085, Texas 9900 and RCA 1802, all on single board computers. PET, TRS-80, APPLE II and AIM-65 microcomputers are available.

A Course in Microcomputers for Control Applications

C. G. Brokus pp. 186-187

S100 based microcomputers are used to allow control experiments to be undertaken. S100 boards are adopted so that the microprocessor may be altered at a future date.

Teaching the Microcomputer to the Two-Plus-Two Technologist

L. P. Sheets pp. 188-191

S100 boards using Intel 8080 microprocessors. An emulator is available that generates 8080 signals at very low speed to enable the student to examine the system using conventional tools.

Computer Technology-Hardware/Software Approach

R. Gaonkar pp. 191-195

The structure of microprocessor related knowledge is explored.

'Microprocessors and Education'

Computer, Vol. 10 No. 1 January 1977

MUMS- A Reconfigurable Microprocessor Architecture

M. Faiman, A.C. Weaver, R.W. Catlin pp. 11-17

A standard bus, with standard input/output devices, can be used by a variety of microprocessors that have been interfaced to the bus.

A Microprocessor Chip Designed with the User in Mind

W.E. Wickes pp. 18-22

A manufacturer discusses the factors that influence the design of a microprocessor.

Automated Generation of Cross-System Software for Microcomputers

G.R. Johnson, R.A. Mueller pp. 23-31

A skeletal simulator and assembler package is presented, designed to allow the rapid introduction of support facilities for new microprocessors.

Microcomputers in the Computer Engineering Curriculum

J.F. Wakerly, E.J. McCluskey pp. 32-38

The authors examine the main areas in which microprocessor education differs from current computer education.

A Microprocessor Laboratory for a University Environment

B.J. Carey pp. 40-46

The author proposes a four tier laboratory based on PDP 11/10 providing support for work on PDP 11/20. Following these stages, the students assembles pre-built modules and finally builds a microcomputer using bread-board techniques.

Teaching Microcomputer Interfacing to Non-Electrical Engineers

P.R. Rony, D.G. Larsen pp. 53-57

Single board computers based on the Intel 8080 are coonfigured so that the student may construct a working system without soldering.

Special Feature: Fault Diagnosis of Microprocessor Systems

V.P. Srini pp. 60-65

Methods for determining the location of a fault causing the malfunctioning of a microprocessor system are described.

'Microprocessor Education'

Euromicro Newsletter, Vol. 3 No. 2 April 1977

Microprocessor Education

J.D. Nicoud pp. 3-4

The results of a survey into the state of microprocessor education are briefly discussed.

Microcomputer Education in the U.S.

M.E. Sloan pp. 5-8

The state of microprocessor education in the U.S. is given.

Report on the DISE Workshop on Microprocessor and Education

Y. Chu pp. 8-11

Conference report on microprocessor education techniques.

Microcomputer Education in an Industrial Environment

R.T. Boute pp. 12-15

A scaled down computer course is presented as the basis of a microprocessor course. The dangers of the apparent ease with which simple programs can be written in assembler are pointed out.

Hardware Engineering and Software Engineering

B.R. Gaines pp. 16-21

The importance on an integrated hardware/software approach to the design of systems is explored.

Microprocessors: New Devices - Classical Problems

J. Katzenelson. M. Werner pp. 22-25

The problems involved in introducing both students and practicing engineers to the microprocessor are discussed. The microprocessor is seen as the basis for all future introductory computer science courses.

The Microcomputer and Computer Science Education

J.M. Hemphill pp. 26-27

Intel 8080 based multi-board computers are used to introduce students to

computer science.

Pro-Log Microprocessor Design Course

J. McDonald pp. 28-30

An industrial design course using Intel 8080 single board computers for logic replacement is presented. The students are encouraged to work in machine code.

Microprocessor Studies at Claude Bernard University

R. Parchoux, P. Desgoutte, J.P. Comte pp. 31-33

Students are presented with 'hardened' microprocessor components to be wired together into a minimal system.

An Experience on Microprocessor Education

J. Figueras, A. Alabau pp. 34-38

A combination of development systems, minicomputers, single board computers and in-house modules based on a variety of microprocessor families provide a wide range of facilities.

Information Processing, Automation and Microprocessor Education

R. Husson, C. Goeldel, J.P. Thomesse, P. Nonn pp. 39-45

An M6800 based in-house system offering machine code programming is discussed. An emulator and high level languages are being imlemented.

Teaching Microprocessors at University

E.L. Dagless, J.S. Mason pp. 46-50

Single board computers based on the 8008 are used with A/D, D/A converters to allow experiments to be undertaken. Other systems available are

based on the M6800 and IMP 16 microprocessor families.

Microprocessor Workshop and Educational Laboratory

J.Z. Loeb, D. Dack pp. 51-56

A minicomputer provides software support for Intel 8080 based microcomputers, each having a switch/light front panel giving access to all bus signals.

Teaching Microcomputer Interfacing to Non-Electrical Engineers

P.R. Rony, D.G. Larsen pp. 57-63

Reprint of 'Computer' article referenced previously.

Microprocessors in a University Electronic Laboratory

G. Conte, D. Del Corso, M. Giordana pp. 63-65

A standard bus is used to allow Intel 8080, Motorola M6800, Signetics 2650 and Zilog Z80 devices to conect to common input/output modules. Access to the microcomputers is via hexadecimal keypad. Software support is provided by a PDP 11/10.

On the Design of a Micro-Processor-Education-System

N. Hojka, K. Judmann pp. 66-68

The Signetics 2650 is used as the basis for a single board computer with a switch/light interface to the user.

Equipment for a Microprocessor Laboratory

J.D. Nicoud pp. 69-72

A standard bus and assembler language are used to introduce concepts, whilst a range of eight-bit microprocessor modules may be interconnected with

memory and input/output modules to provide a complete microcomputer. Software support is available on a range of minicomputers.

Behaviour Display System for a Microprogrammed Processor

S. Monchaud, J.F. Pedenon, B. Lemaire pp. 73-75

A block diagram of a microprogrammed processor is animated by sequenced lights, controlled by a microprocessor of the type displayed.

Software Model of the M6800 Microprocessor

K. Muhlemann pp. 76-79

An M6800 microprocessor simulates an M6800, allowing the ready examination of the internal state of the machine on a monitor.

CASS8: A cross Assembler suitable for most of the 8 Bit Microprocessors

J. Tiberghien, J. Taeymans pp. 80-84

The dangers of machine code are stated and a table driven cross-assembler is described.

Comparison of Microprocessors: A Graphical Approach

D. Aspinall, M.H. Barton, E.L. Dagless pp. 84-92

Graphical methods of describing the facilites offered by particular microprocessors are presented. The result is an easier means of selecting the correct microprocessor for a particular task.

13. Acknowledgements

The Author wishes to express his thanks to :

Mr. A.R. Daniels, for his supervision, assistance and encouragement;

Professor J.F. Eastham, for the use of School of

Electrical Engineering Resources;

The Undergraduate students who suffered the quirks of the described

unit and added to the completeness of the system;

All members of the School,

for discussions and their valued comments;

Mrs. J Newman for the diagrams;

And my wife, for assistance with the typing.

14. Appendices

These appendices will be found in the wallet attached to the inside back cover of the thesis and consist of:

Appendix 1. Full circuit diagrams for boards one and two of the supervisory microprocessor, interface and processor boards for the M6800 and interface and processor boards for the M68000.

Appendix 2. Pre-launch publicity information sheet describing the commercial version of the complete system.

15. Tables and Figures

Control Latch Contents Bit Number 7-------0	Command Mnemenoic	Action
XXXXXXX?	HALT	State of Bit 0 directly fed to HALT line
XXXXXX?X	RESET	State of Bit 1 is directly fed to $\overline{\text{RESET}}$ line
000XXXXX	UPLD	Current data byte is loaded into most significant byte of interface address latch
001XXXXX	DNLD	As UPLD but lower byte
010XXXXX	INCADD	Increment contents of Address Latch
011XXXXX	DECADD	Decrement contents of Address Latch
100XXXXX	VMAC	Controls state of modified VMA line
101XXXXX	SS	Single steps a halted M6800
110XXXXX	IRS	Sets bit in interface status latch for polling
111XXXXX	-	No operation
XXX000XX	INSTLD	Load Interface Instruction latch
XXX001XX	R/W	Value for read/write line during DMA
XXX010XX	INIT	Initialize interface
XXX011XX	DPSH	Push instruction onto data bus (use with SS)
XXX100XX	NMI	Pulses $\overline{\text{NMI}}$ line of M6800
XXX101XX	IRQ	Pulses $\overline{\text{IRQ}}$ line of M6800
XXX111XX	-	No operation

Note, a data byte is transmitted simultaneously with the command byte and contains the value required by command.

Table 1. Commands available to PDP 11 using an externally controlled DMA channel to operate an M6800

Signals on Prom Address lines					Read / Write / Neither	by	Master / Slave	to	Board1 / Board2 / Beyond	Action by: Board2/ Board1 buffer	Action by: Board2/ Interface buffer
$\overline{RD}$	$\overline{MWR}$	$\overline{ONCARD}$	$\overline{BACK'}$	$\overline{BEYOND}$						(notes)	
0	0	X	X	X	X		X		!	(1)	
0	1	0	0	0	R		S		!		
0	1	0	0	1	R		S		2		towards i/f
0	1	0	1	0	R		S		B		
0	1	0	1	1	R		S		1	towards i/f	
0	1	1	0	0	R		M		!		
0	1	1	0	1	R		M		2	(2) towards Z80	towards i/f
0	1	1	1	0	R		M		B	towards Z80	towards Z80
0	1	1	1	1	R	by	M	to	1	(2) towards i/f	towards i/f
1	0	0	0	0	W		S		!		
1	0	0	0	1	W		S		2		towards Z80
1	0	0	1	0	W		S		B		
1	0	0	1	1	W		S		1	towards Z80	towards Z80
1	0	1	0	0	W		M		!		
1	0	1	0	1	W		M		2	towards i/f	
1	0	1	1	0	W		M		B	towards i/f	towards i/f
1	0	1	1	1	W		M		1		
1	1	X	X	X	N		X			(3)	

! Indicates a state that cannot occur unless the circuit is faulty

X Indicates a "Don't Care"

Notes

1) $\overline{RD}$ and $\overline{MWR}$ cannot fall together, buffers will be disabled if they should.

2) The contents of the data bus are broadcast through the system for the benefit of Z80 peripherals monitoring the instruction stream for a 'RETI'.

3) The buffers are disabled until a direction is established.

Table 2. Response of the Board 2 Data Bus Buffers to Requests

Signals on Prom Address lines							
$\overline{\text{ONCARDIO}}$		A1		Input/Output		State	
$\overline{\text{BACK}}$	A2		A0	Port	Selected		
					(Notes)		
0	0	X	X	X	—	—	
0	1	X	X	X	—	(1)	Slave Accessing Memory
1	0	0	0	0	40	(2)	Up-dating Master Address RAM
1	0	0	0	1	41	(3)	Up-dating Master Attribute RAM
1	0	0	1	0	42	(4)	Up-dating Slave Attribute RAM
1	0	0	1	1	43	(5)	Up-dating Slave Address RAM
1	0	1	0	0	44	(6)	Up-dating Page Select Register
1	0	1	0	1	—	—	
1	0	1	1	0	—	—	
1	0	1	1	1	—	—	
1	1	X	X	X	—	(7)	Master Accessing Memory

X Indicates a "Don't Care"

Notes

1) Enable Slave Address and Attribute RAMs

2) Enable Master Address RAM and Address RAM output buffer

3) Enable Master Attribute RAM and Attribute RAM output buffer

4) Enable Slave Attribute RAM and Attribute RAM output buffer

5) Enable Slave Address RAM and Address RAM output buffer

6) Enable Page Select Register

7) Enable Master Address and Attribute RAMs

Table 3. Board 2 Memory Management Control PROM Specification

Pin Number	Row a		Row c
1	Ground		Ground
2	+12V		+12V
3			
4			
5			
6			
7			
8			
9	TxData	Asynchronous	
10	$\overline{RTS}$	Interface	
11	TxClock	Communications	
12	RxClock	Adapter	
13	RxData	Lines	
14	$\overline{DCD}$		
15	$\overline{CTS}$		
16			
17			
18			
19	CA2	Parallel	CB2
20	CA1	Interface	CB1
21	PA7	Adapter	PB7
22	PA6	Data Ports	PB6
23	PA5	and	PB5
24	PA4	Control	PB4
25	PA3	Lines	PB3
26	PA2		PB2
27	PA1		PA1
28	PA0		PB0
29			
30			
31	-5V		-5V
32	+5V		+5V

(All unmarked lines are unused)

Table 4. M6800 Board, J2 Connector, Minimal Input/output Signals

Address (in hexadecimal)	Memory Decoded
0000 to 07FE	Vector Space (RAM1 or ROM1)
0800 to 0FFE	ROM1
1000 to 17FE	ROM2
1800 to 1FFE	RAM1
2000 to 27FE	RAM2
2800 to 2FFE	Input/Output
3000 to 3FFE	No memory present (but $\overline{\text{DTACK}}$ generated)
4000 to 47FE	Interface Board Shared Memory
4800 to 7FFE	No memory present (but $\overline{\text{DTACK}}$ generated)
8000 to FFFE	Z80 (via interface card)
10000 to FFFFE	For expansion ($\overline{\text{DTACK}}$ not generated)

Note: The M68000 generates a 24-bit address (corresponding to a value of $FFFFFE_{16}$), but as stated in the text, only twenty lines are fed to the backplane due to pinout restrictions.

Table 5. The Motorola M68000 Address Map

$\overline{a} = M/\overline{IO} \cdot ALOW \cdot \overline{A13} \cdot \overline{A12} \cdot A11 \cdot \overline{y}$

$+ M/\overline{IO} \cdot ALOW \cdot \overline{A13} \cdot A12 \cdot A11$

$\overline{b} = M/\overline{IO} \cdot ALOW \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{x}$

$+ M/\overline{IO} \cdot ALOW \cdot \overline{A13} \cdot A12 \cdot \overline{A11}$

$+ M/\overline{IO} \cdot ATOP \cdot A13 \cdot A12 \cdot A11 \cdot \overline{x}$

$\overline{c} = M/\overline{IO} \cdot ALOW \cdot \overline{A13} \cdot \overline{A12} \cdot A11 \cdot y$

$+ M/\overline{IO} \cdot ALOW \cdot A13 \cdot \overline{A11}$

$\overline{d} = M/\overline{IO} \cdot ALOW \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11} \cdot x$

$+ M/\overline{IO} \cdot ALOW \cdot A13 \cdot \overline{A12}$

$+ M/\overline{IO} \cdot ATOP \cdot A13 \cdot A12 \cdot A11 \cdot x$

$IOS = \overline{M}/IO \cdot \overline{A15} \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11}$

$\overline{OFBD} = ALOW + M/\overline{IO}.ATOP. A13 \cdot A12 \cdot A11$

Note : $ATOP = A19 \cdot A18 \cdot A17 \cdot A16 \cdot A15 \cdot A14$

$ALOW = \overline{A19} \cdot \overline{A18} \cdot \overline{A17} \cdot \overline{A16} \cdot \overline{A15} \cdot \overline{A14}$

x and y are provided by the address map selection switches

a and b control the selection of the RAM devices

c and d control the selection of the ROM devices

IOS enables the input/output devices whilst $\overline{OFBD}$ controls indicates that the memory requested is not on board.

Table 6. Intel 8086 Processor Board Memory and Input/Output Decode PAL contents

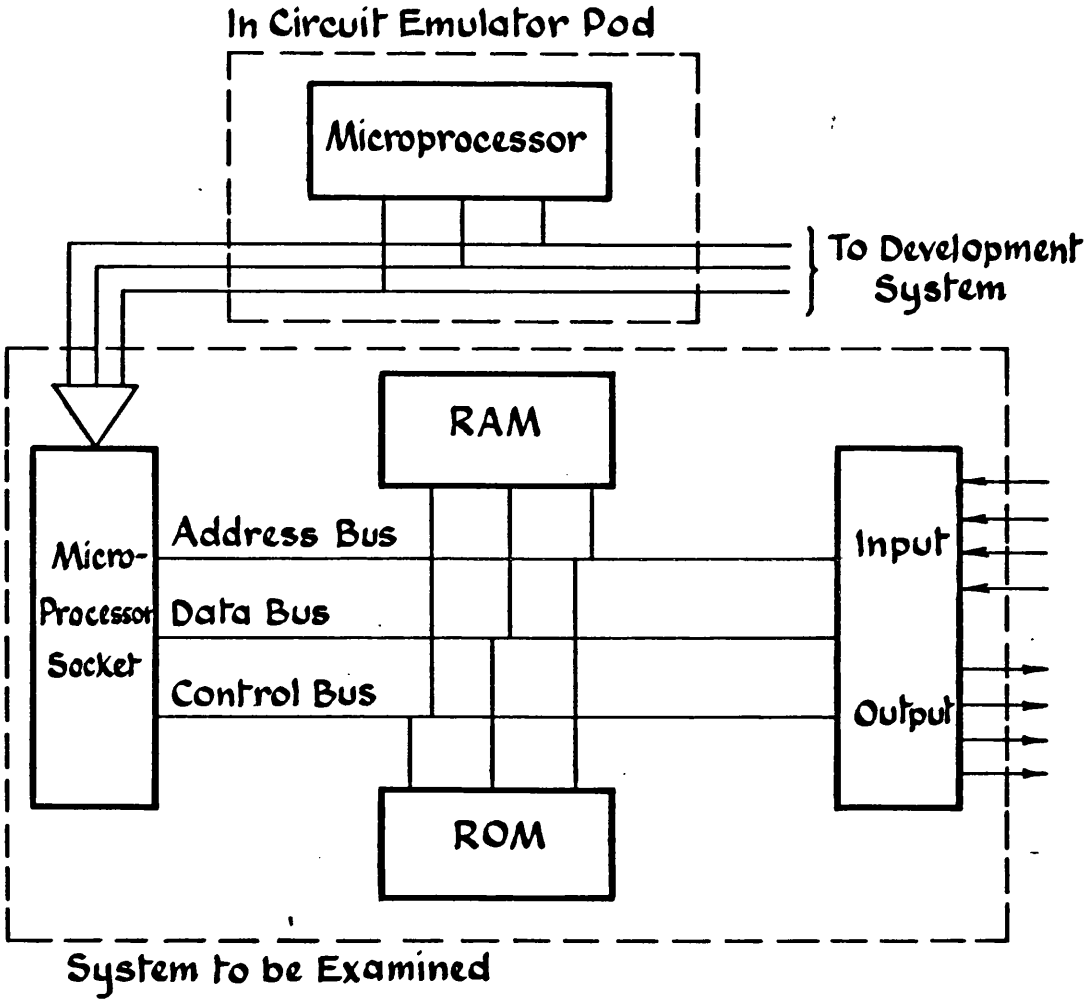


Fig 3.1. Examination of a Microcomputer System by In-Circuit Emulation.

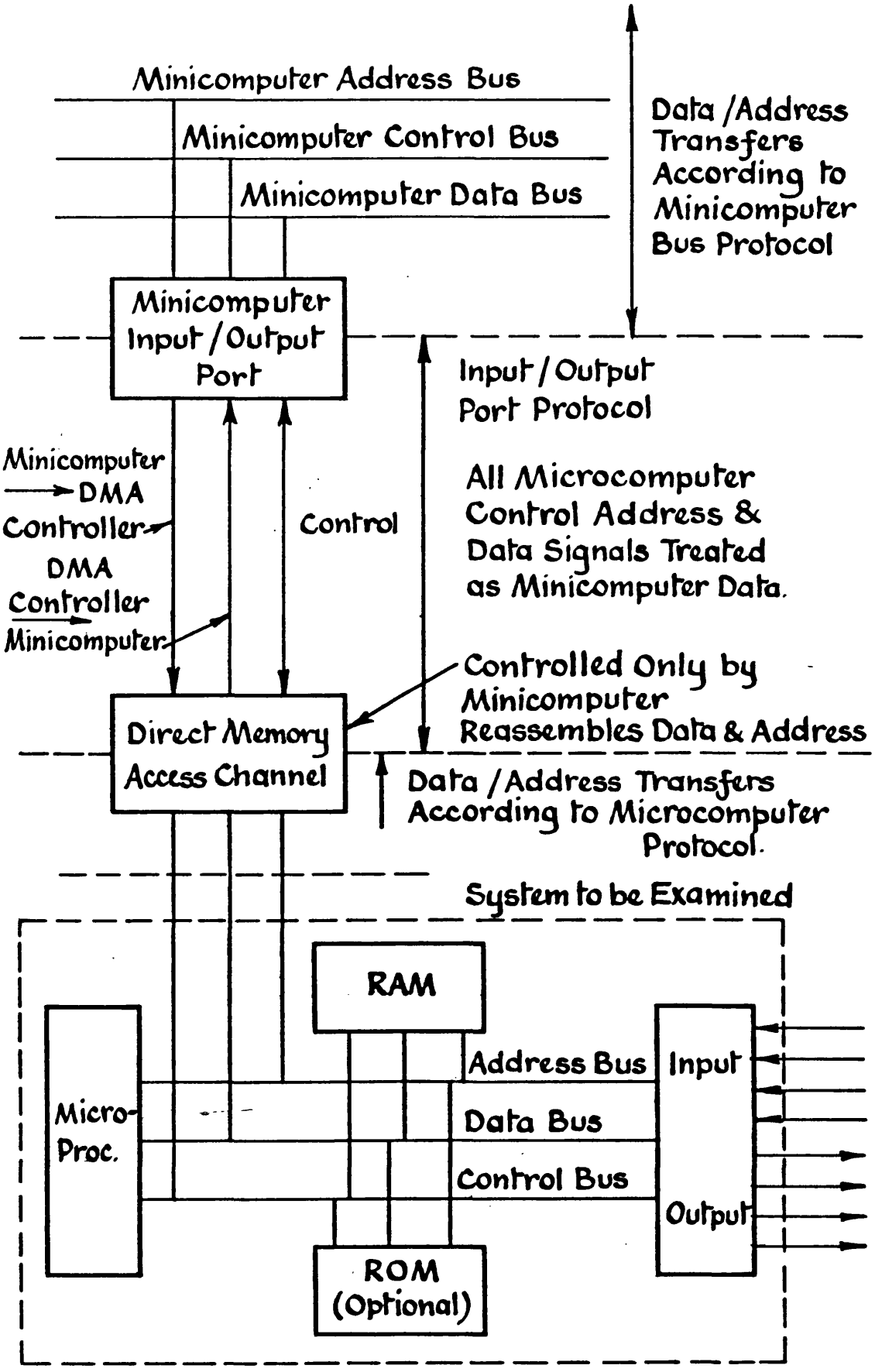


Fig. 3.2 Examination of a Microcomputer by Externally Controlled Direct Memory Access Channel.

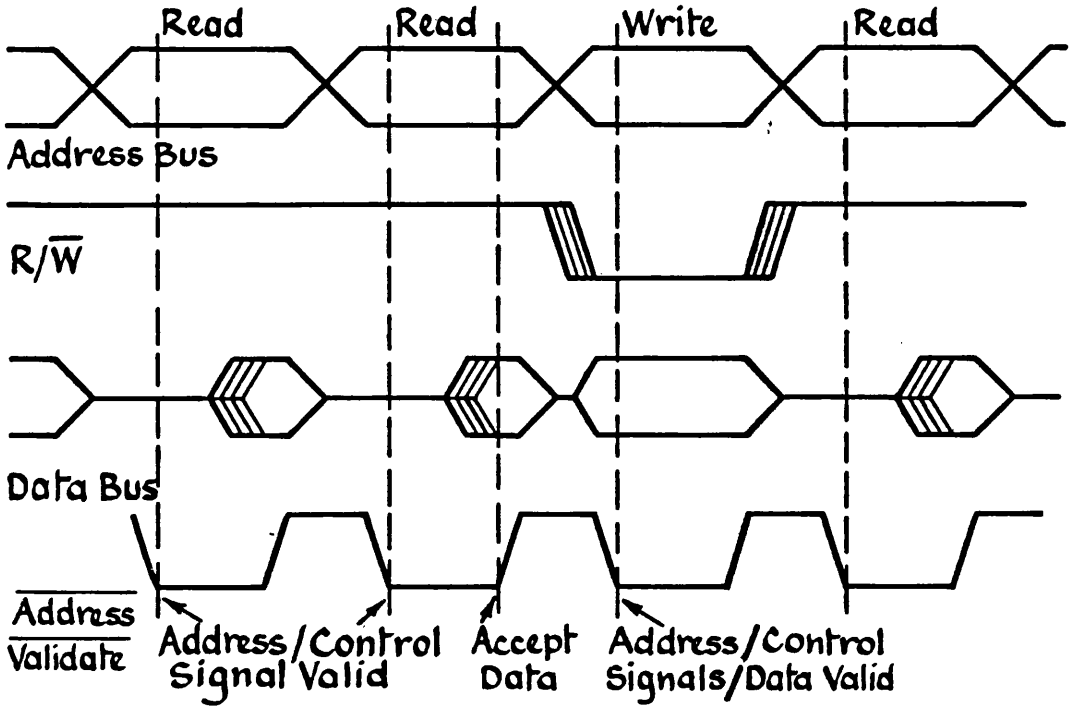


Fig. 4.1 Combined R/$\overline{W}$ & Address Validation Signals Meeting Bus Protocol Requirements.

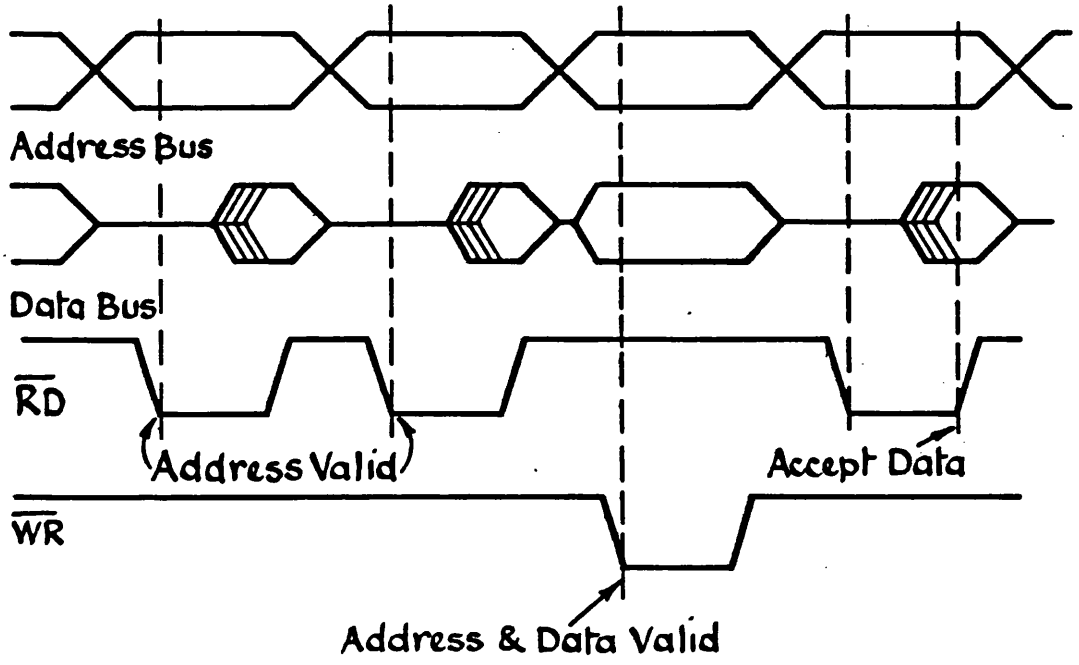
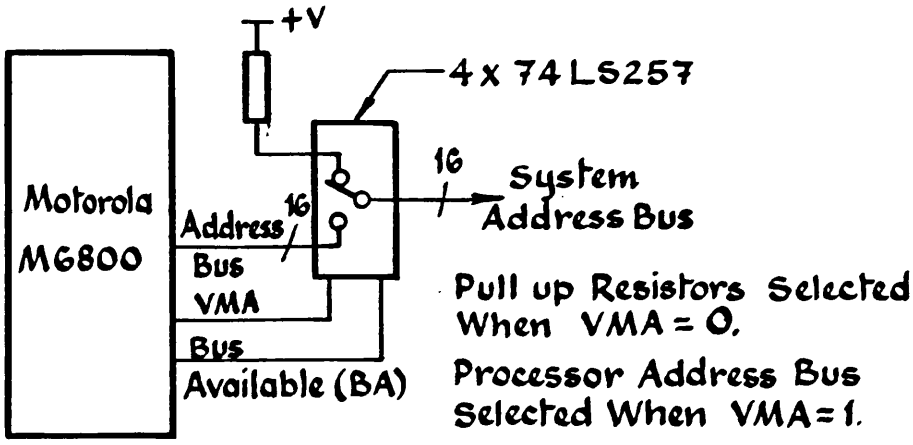


Fig 4.2. Separate Read & Write Signals Providing Address & Data Validation Required for Bus Protocol.



Result: All Invalid Cycles Read Reset Vector ($FFFF_{16}$)

Fig. 4.3 Circuit to Replace Valid Memory Address (VMA) Signal in Motorola M6800 Systems.

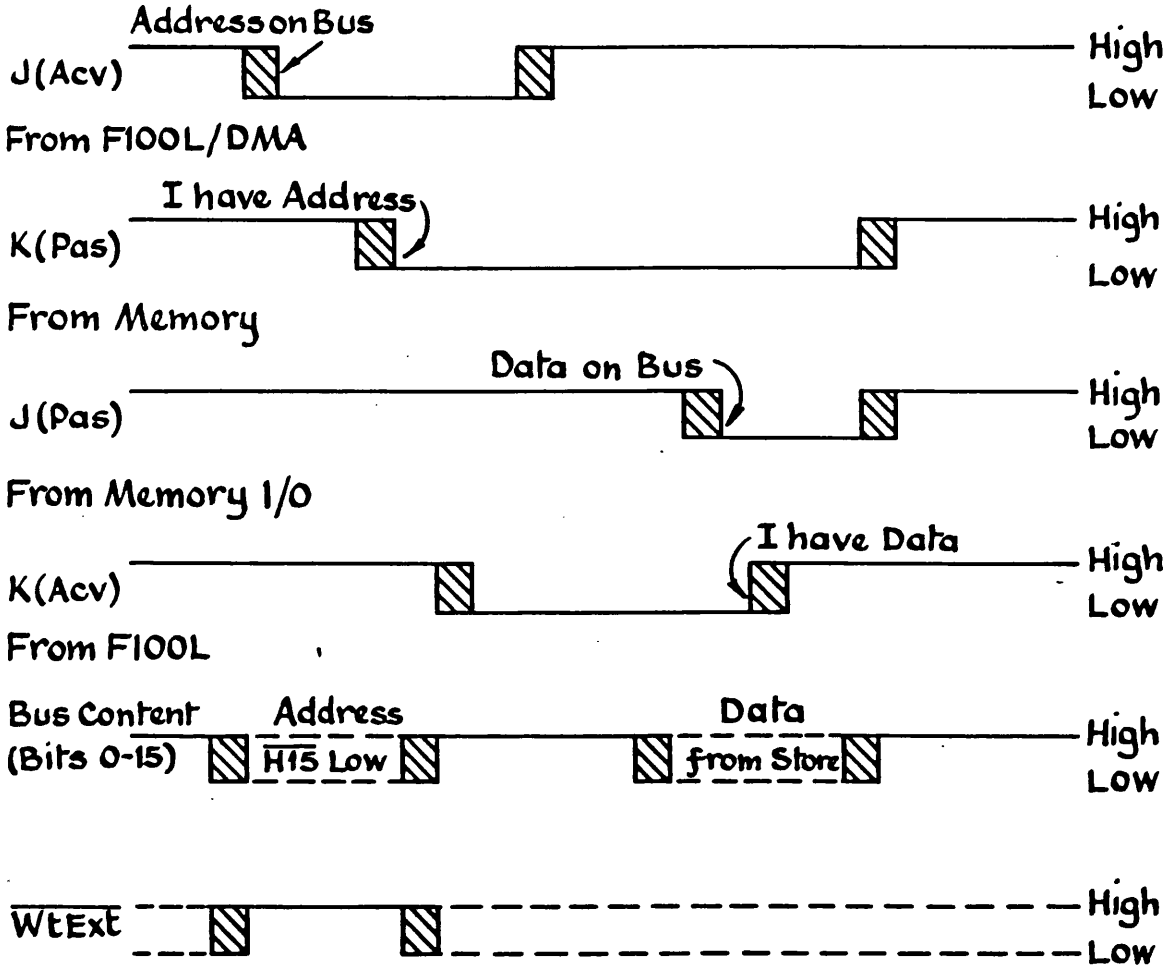


Fig 4.4 F100 Read Only Cycle.

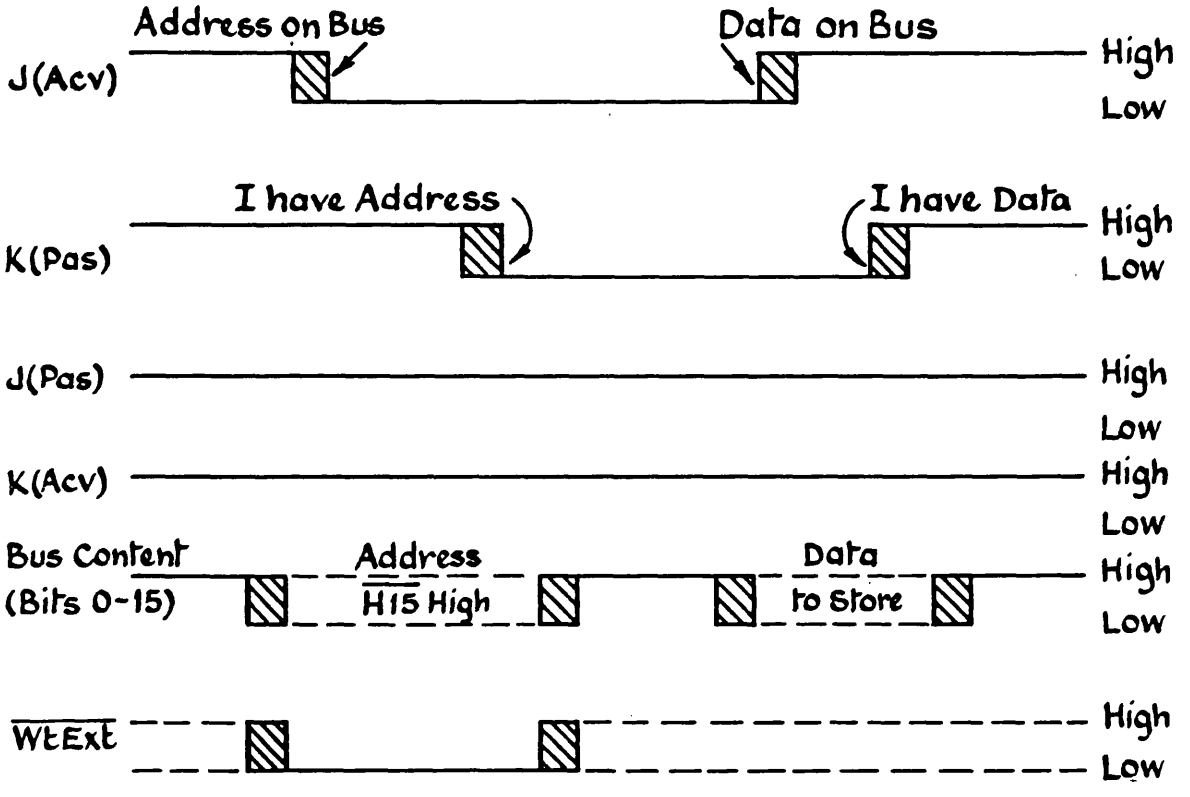


Fig. 4.5. F100 Write Only Cycle

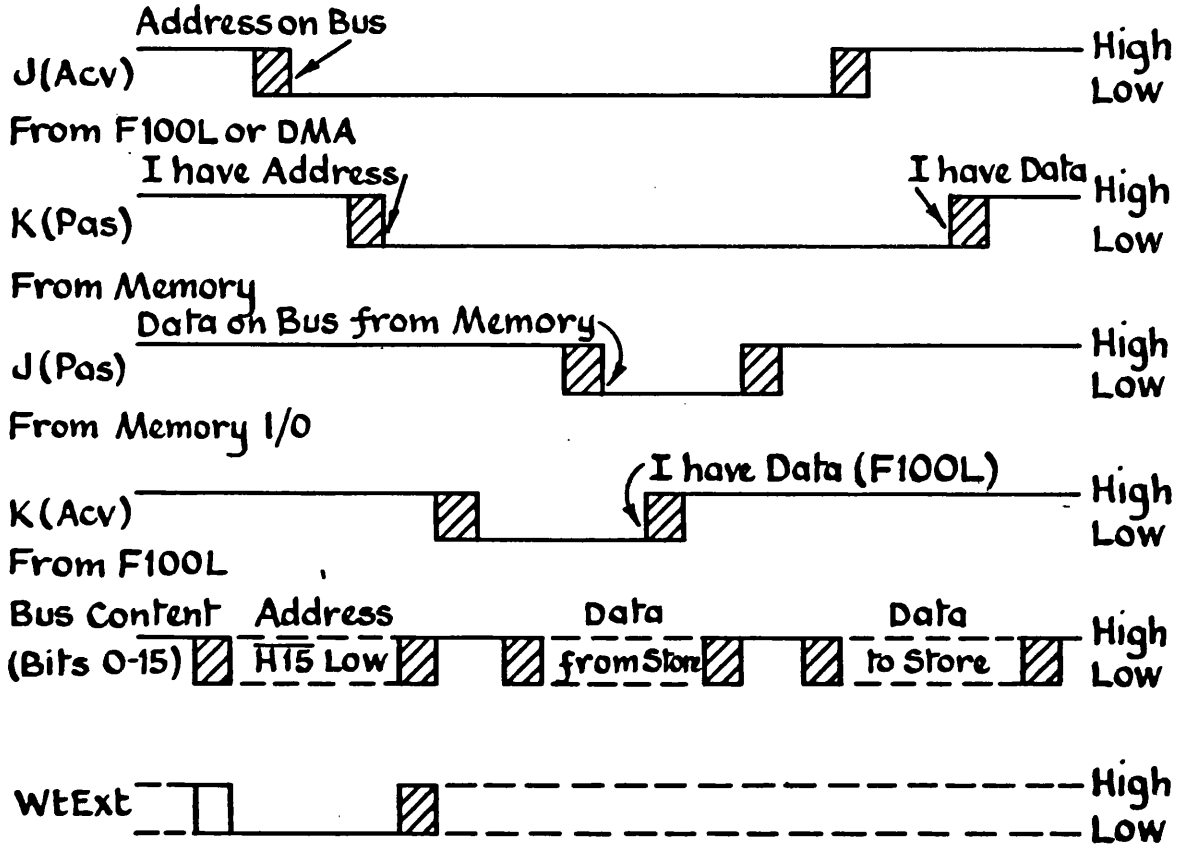


Fig 4.6 F100 Read /Modify/Write

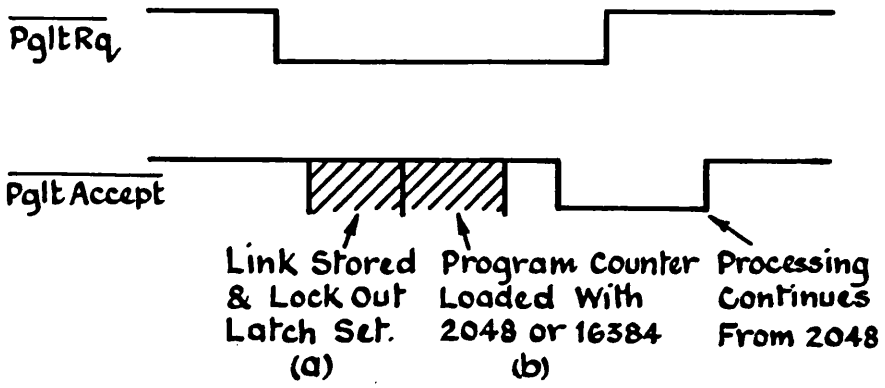


Fig 4.7 The F100-Unvectored Program Interrupt Sequence

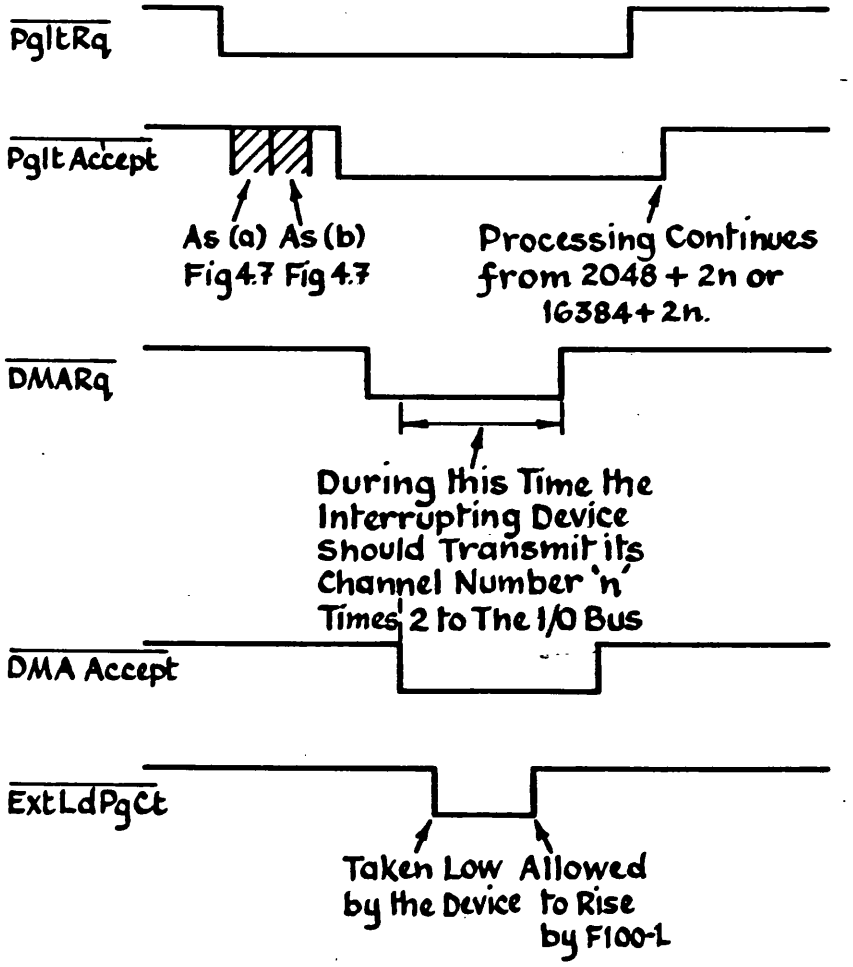
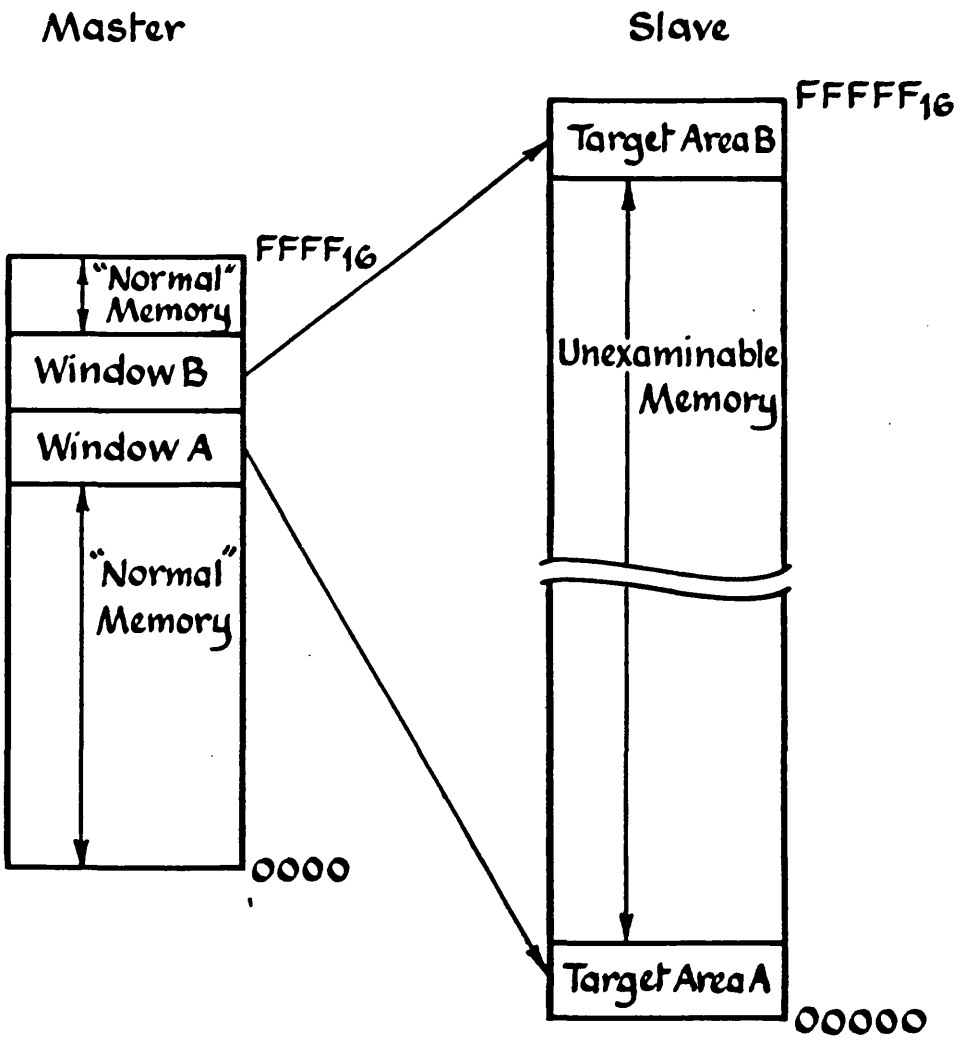


Fig 4.8. The F100 Vectored Program Interrupt Sequence



When Master Accesses Window A or B, Memory Devices in Slave Target Area A or B Respond.

Fig 5.1. Fixed Window Memory Manager

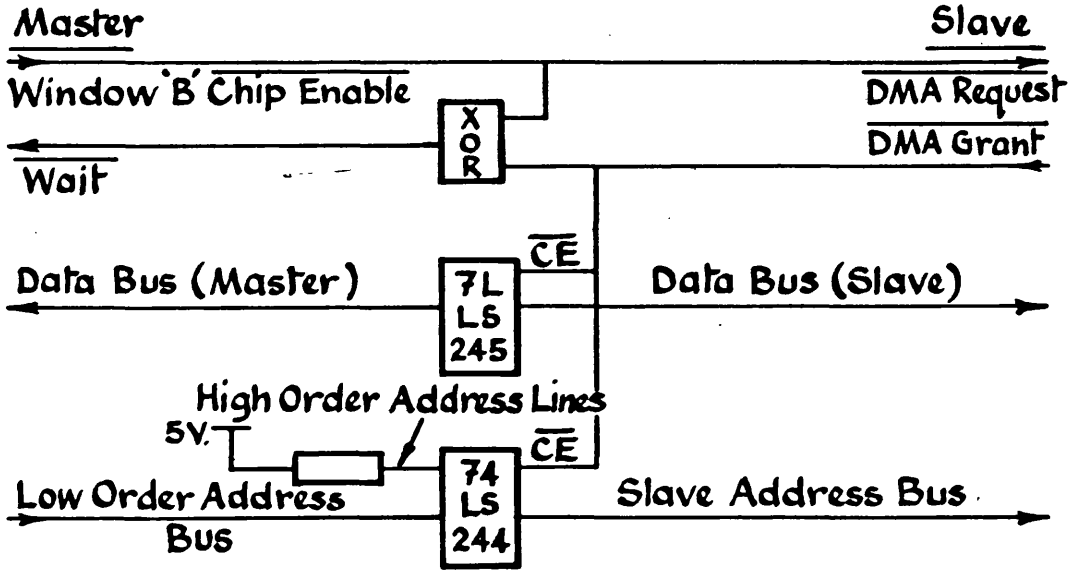


Fig. 5.2 Possible Bus, Fixed Window Memory Manager Implementation (Window B Only).

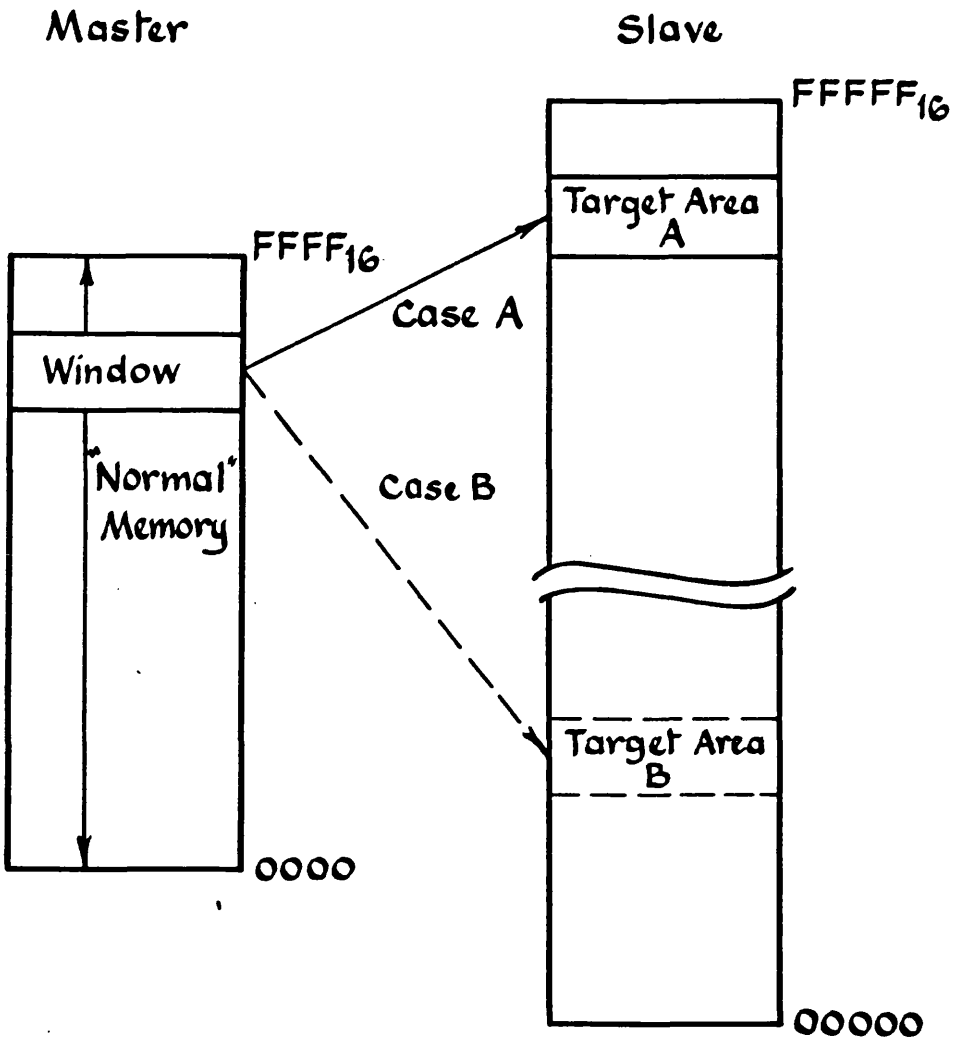


Fig 5.3 Single Window Memory Manger. The Access Shown for Two Possible Latch Valves.

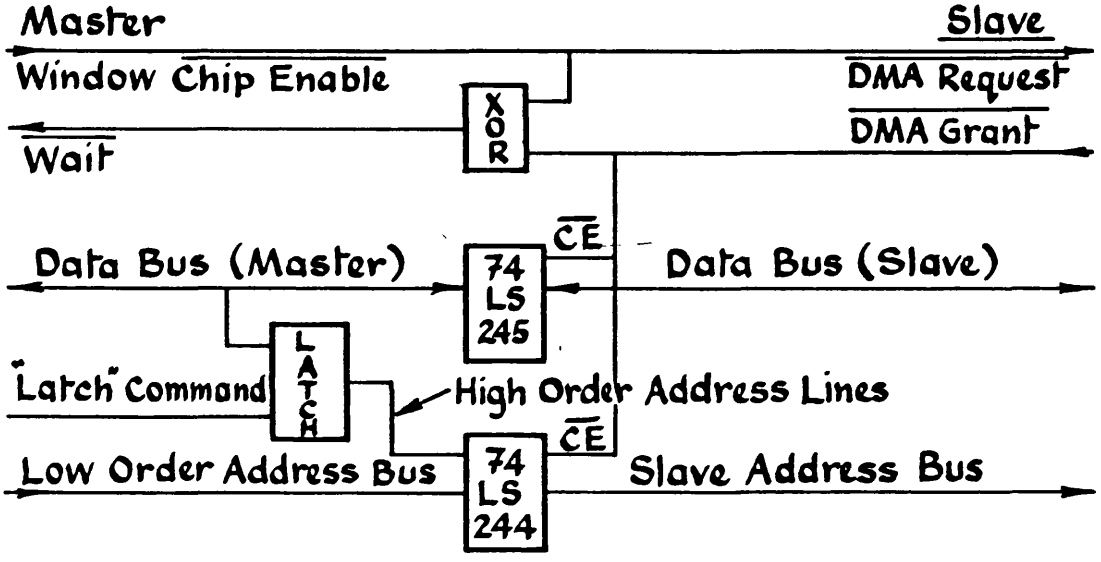


Fig 5.4. Possible Single Window Memory Manager Implementation.

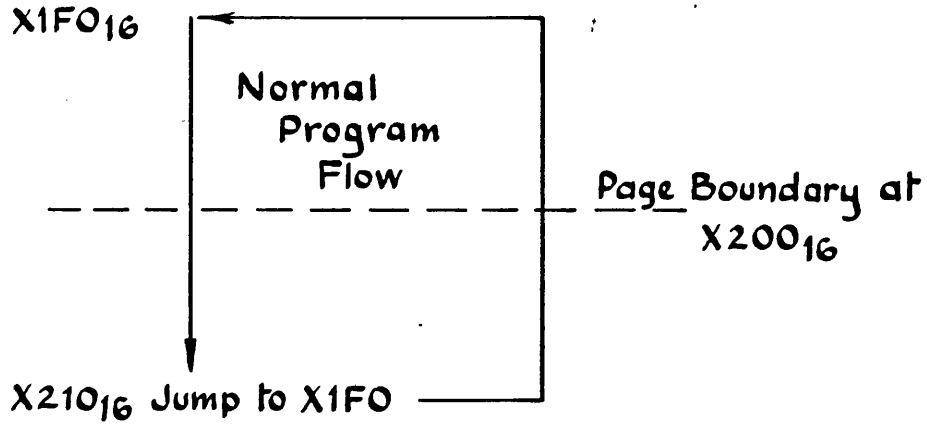


Fig 5.5 Program Loop Across Page Boundary Forces Frequent Changing of Access Latch.

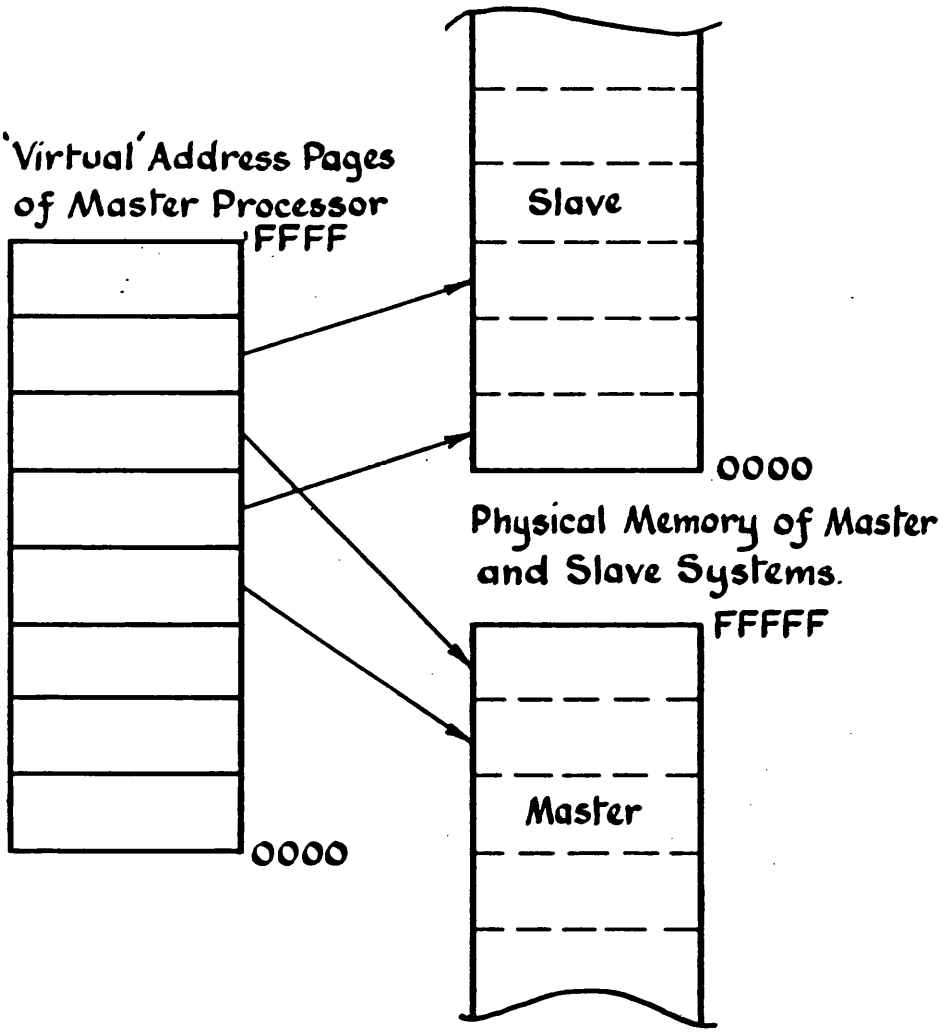


Fig 5.6. Integrated Host / Slave Memory Management All Processor Accesses Undergo Translation to a Physical Address.

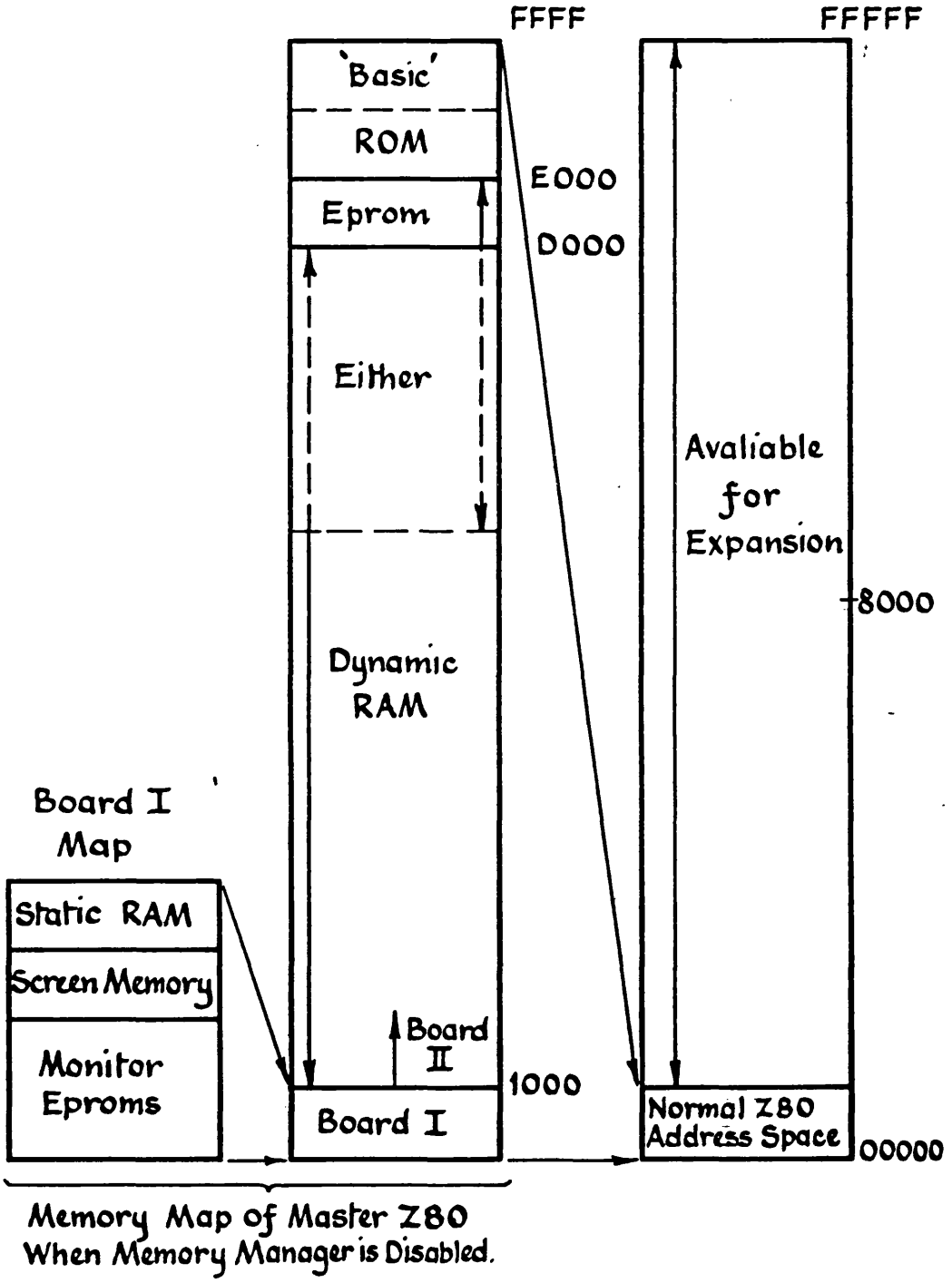


Fig 5.7. Memory Map of Master Z80.

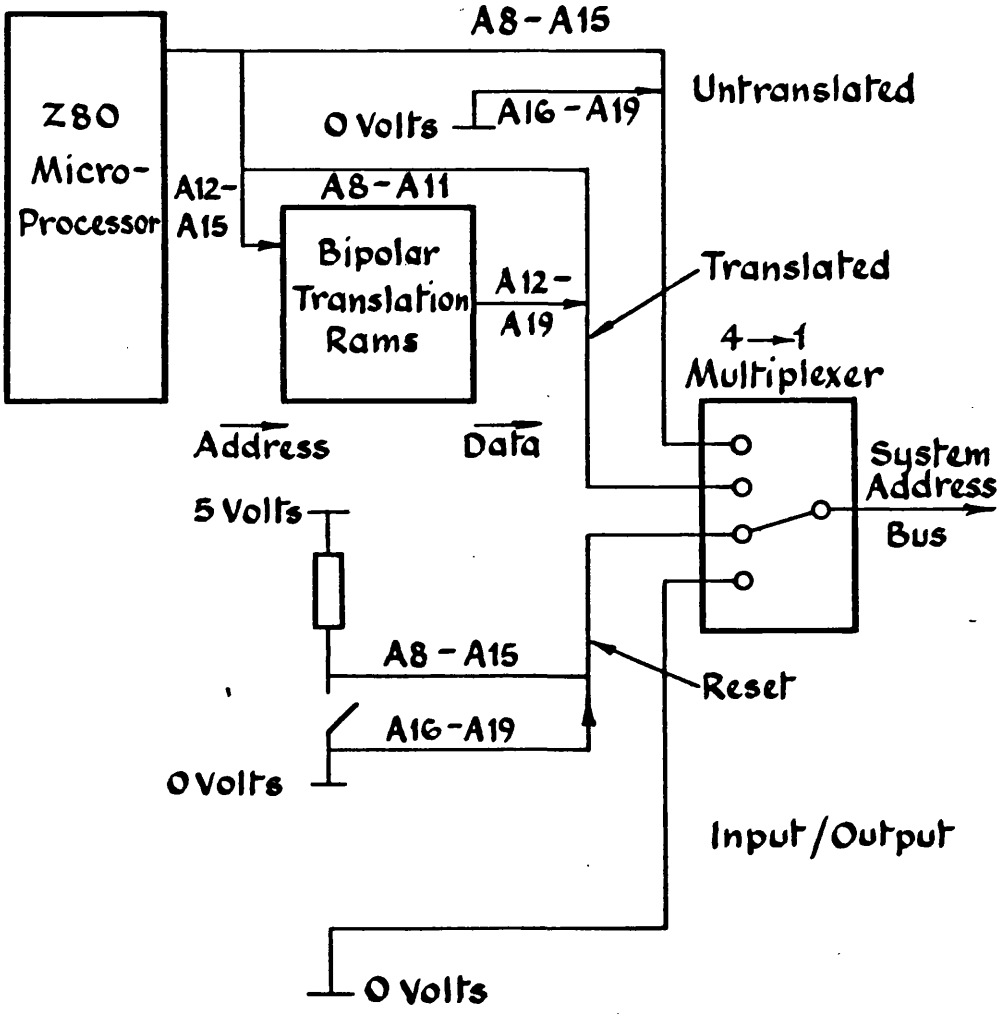


Fig 5.8 Four to One Multiplexers Selecting Address Source to be Fed to System Bus.

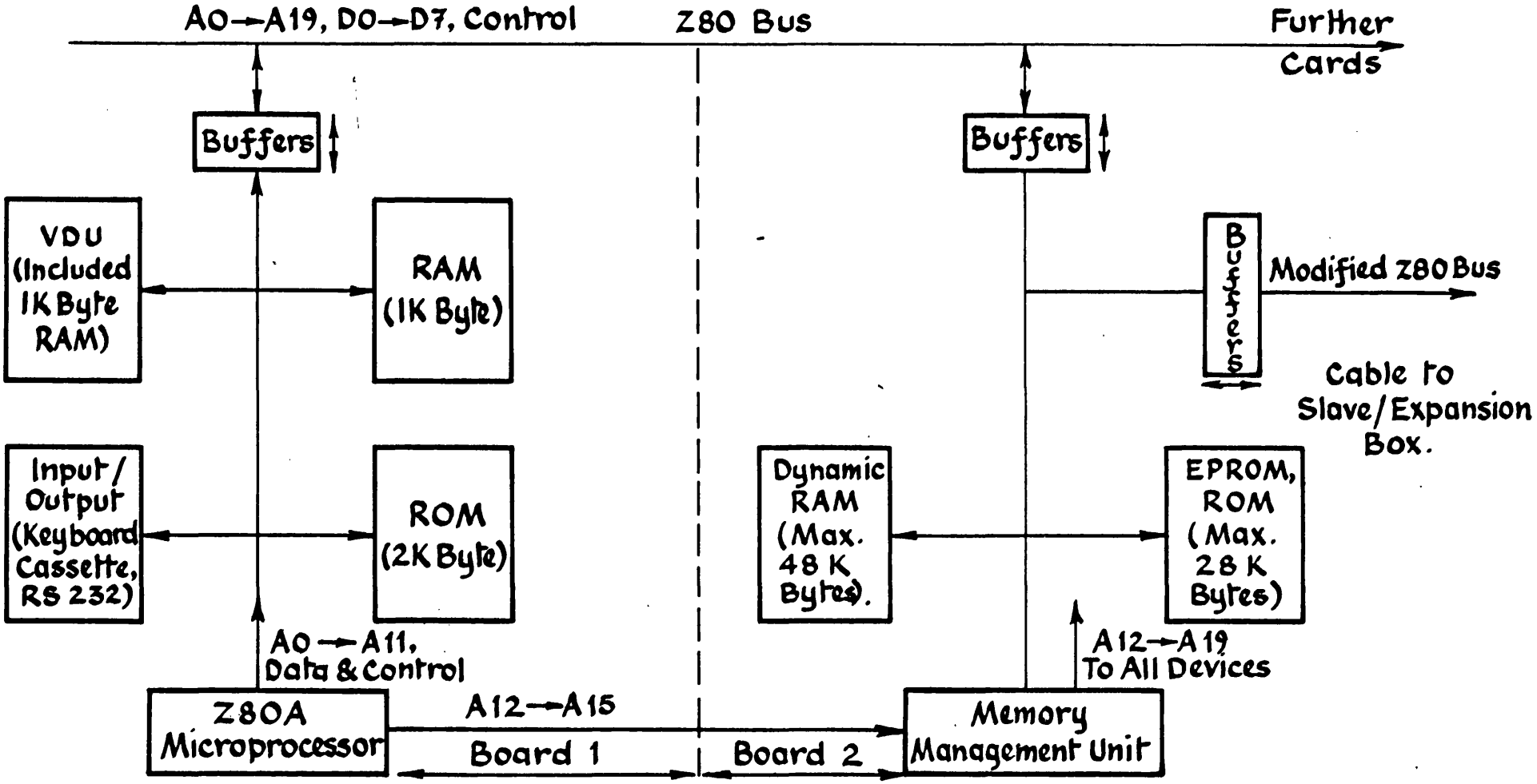


Fig 6.1. Diagram of Supervisory Microprocessor System.

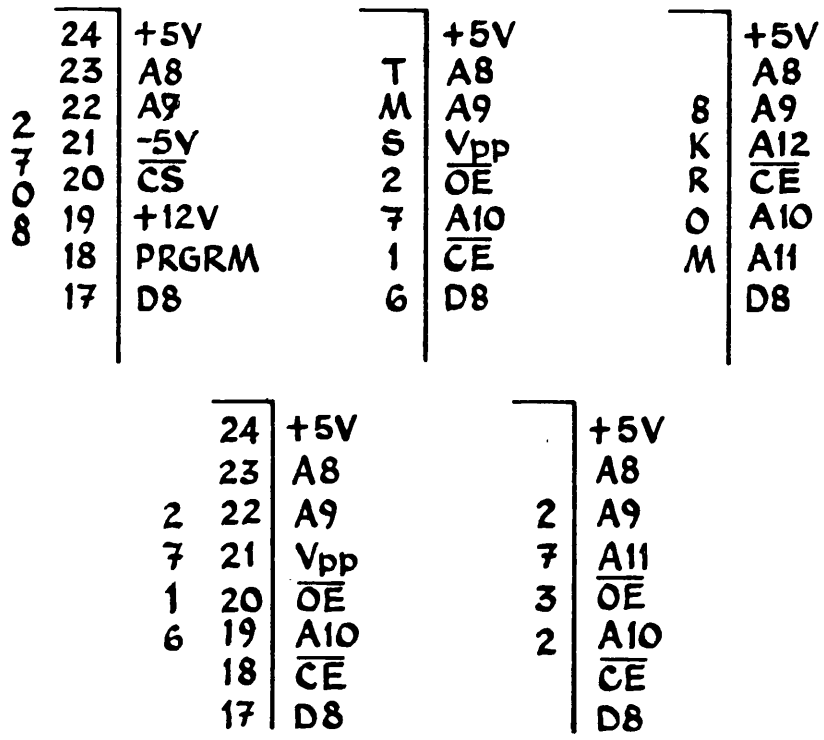


Fig 6.2. Partial Pin-Outs of Eproms & 'Basic' ROM

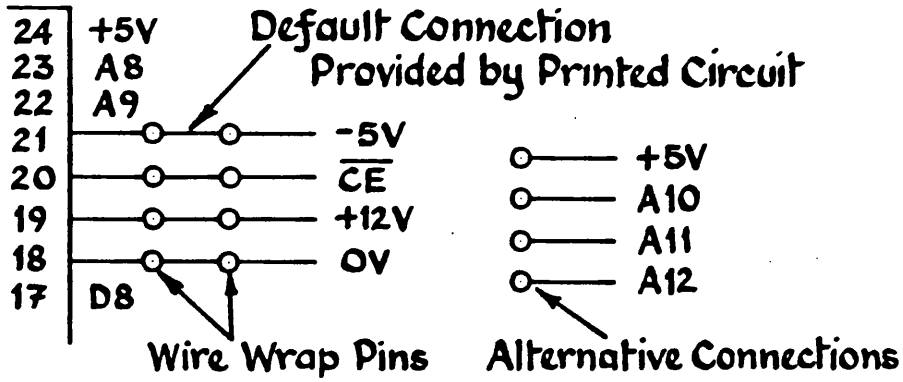


Fig. 6.3 Wire Wrap Links Allowing Any of Above Eproms to be Fitted.

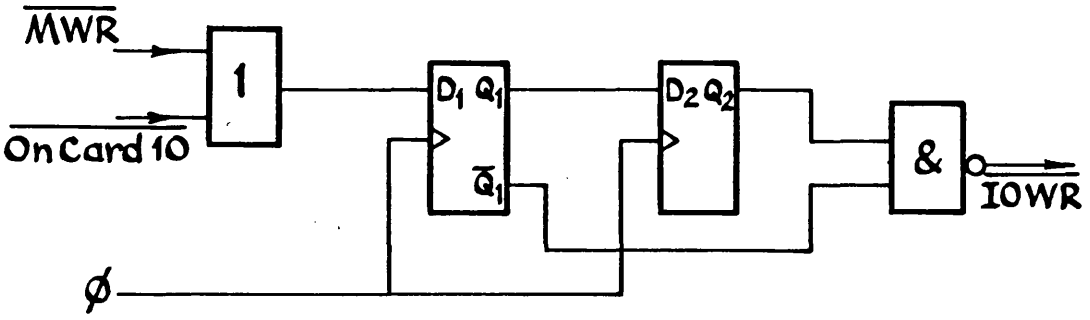


Fig. 6.4. $\overline{IOWR}$ Generation Circuitry

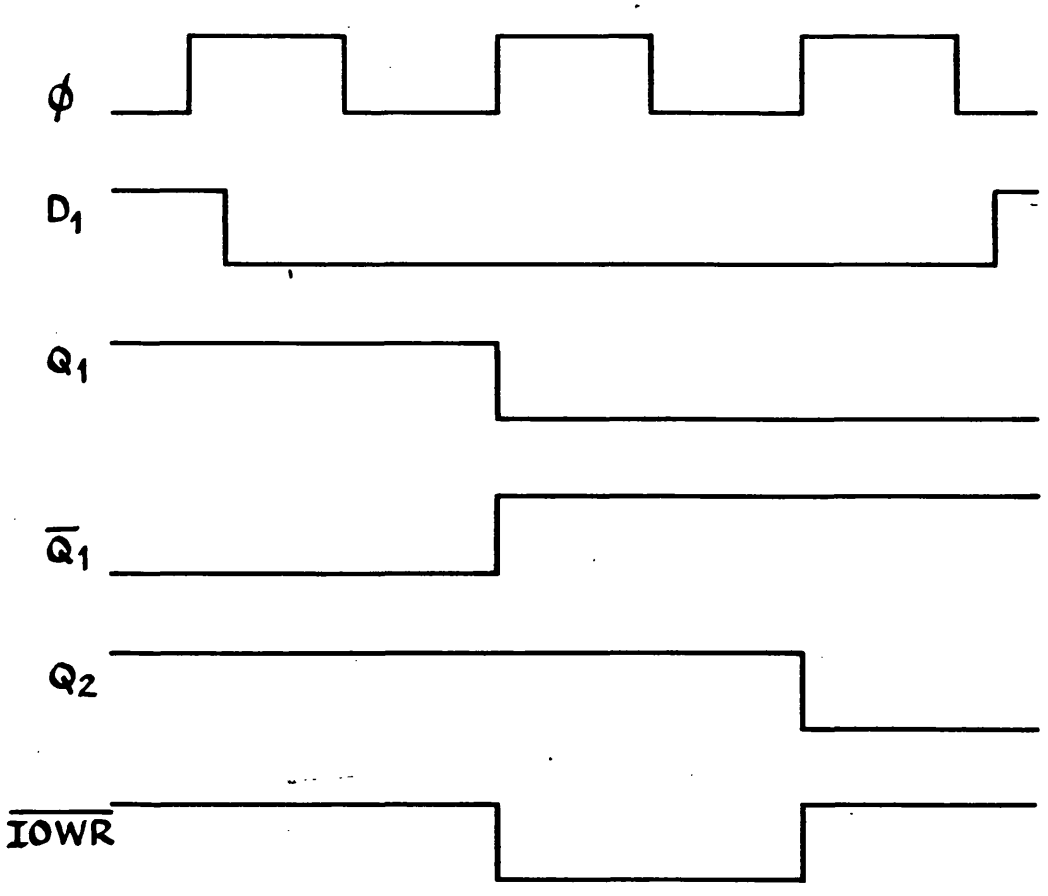


Fig 6.5. $\overline{IOWR}$ Generation Waveforms

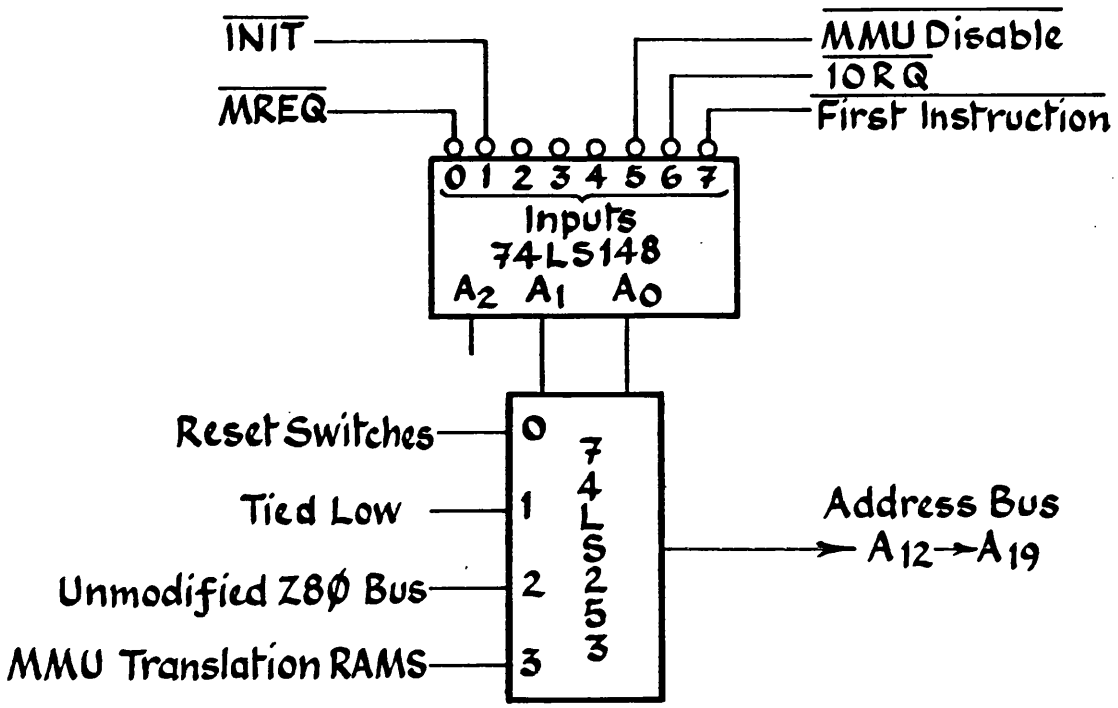


Fig 6.6 Use of 74LS148 To Control Address Source Selection

Input	Signal	74LS148 Output on A1 A0	Data Source Selected
0	$\overline{\text{MREQ}}$	3	Translation RAMS.
1	$\overline{\text{INIT}}$	2	Z80 Bus Lines
2	–	1	Tied Low
3	–	0	Reset Switches
4	–	3	Translation RAMS.
5	$\overline{\text{MMU Disable}}$	2	Z80 Bus Lines
6	$\overline{\text{IORQ}}$	1	Tied Low
7	$\overline{\text{First Instruction}}$	0	Reset Switches

↓ Increasing Priority

Fig 6.7. 74LS148 Signal Assignments.

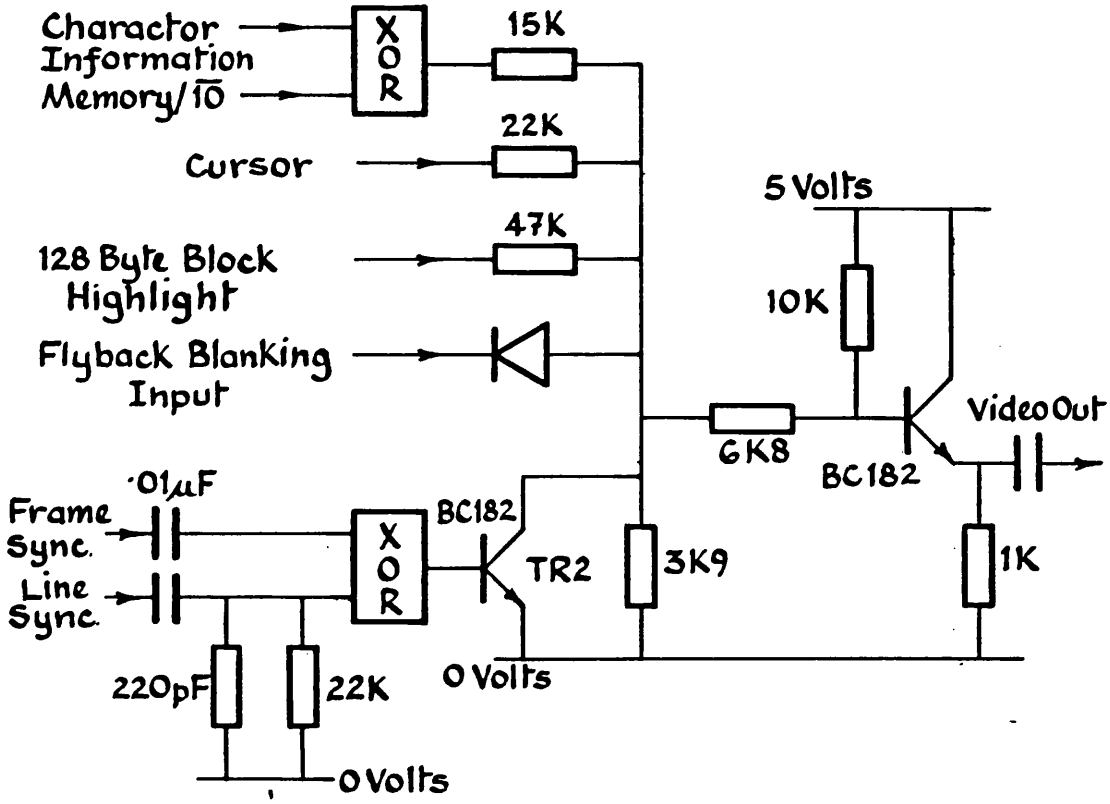


Fig 6.8 "Sofy" Video Summing Circuit

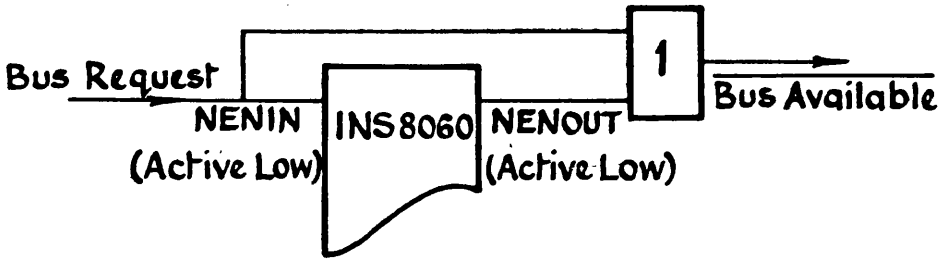
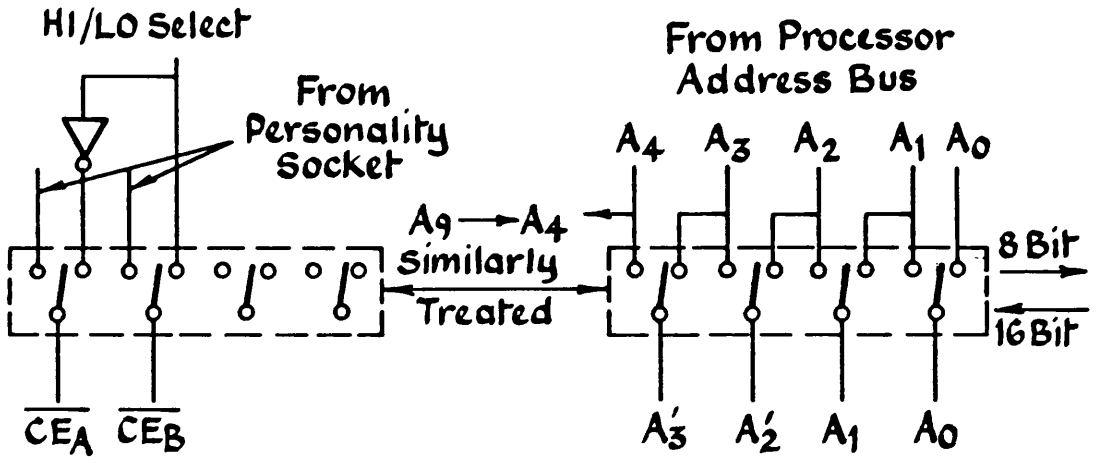


Fig 6.9 The Internal OR Gate of the INS8060 is Augmented by the Faster LSTTL OR Gate to Force the Video Buffers off the Bus More Quickly.



When an External Access is not Being Performed, the Personality Socket Provides $\overline{A\phi}$ & $A\phi$ to $\overline{CE_A}$, $\overline{CE_B}$.

Fig 6.10. Addressing Scheme to Allow "Softy" Buffer Access by Eight and Sixteen Bit Processors.

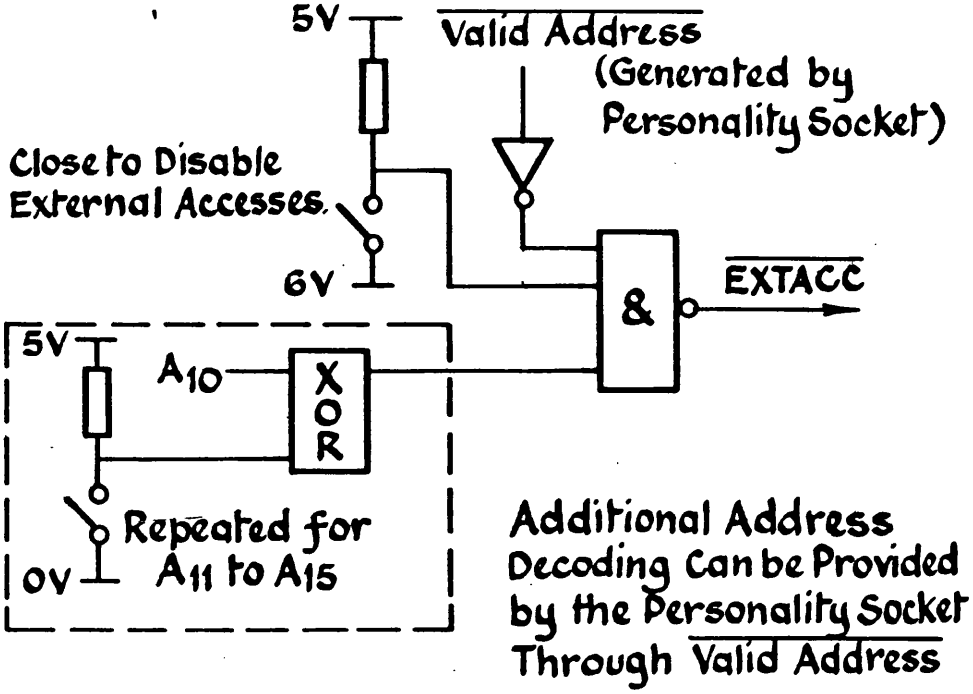


Fig. 6.11. "Softy" Selection Circuit.

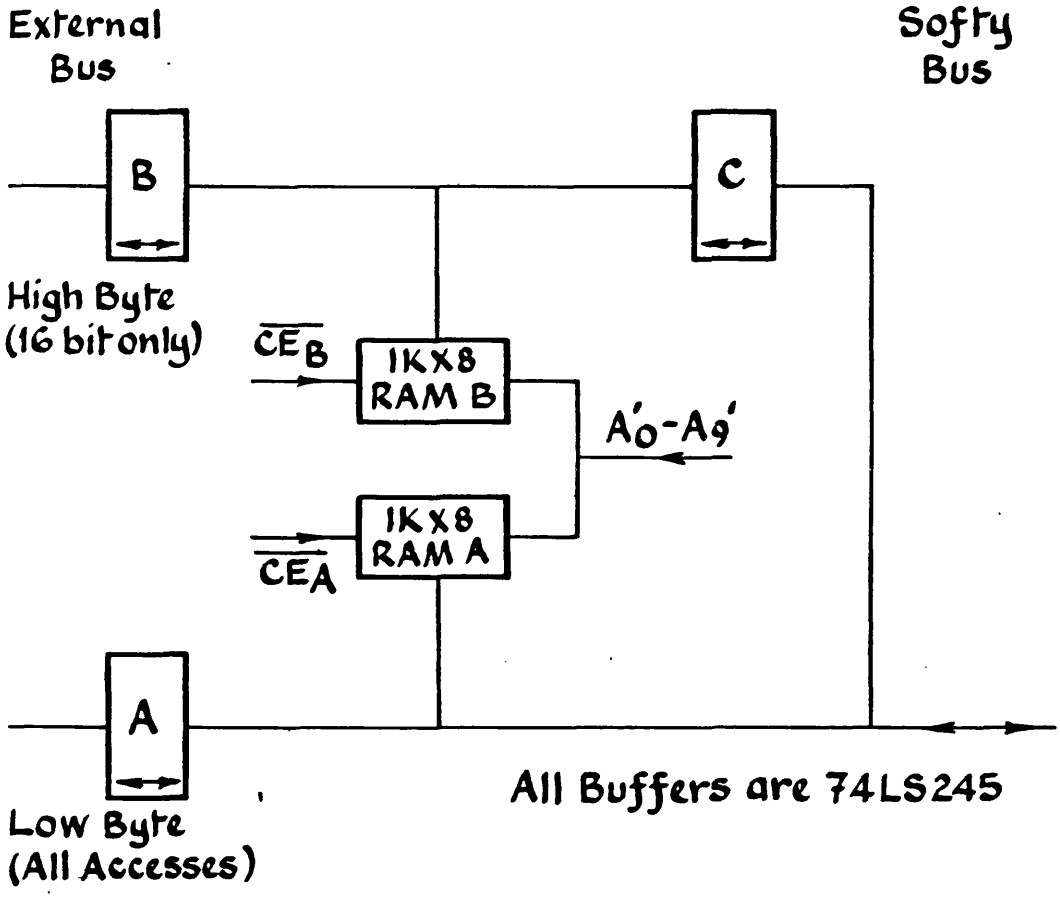


Fig. 6.12. Softy RAM Buffer Layout.

Pin Number	Function		Pin Number	Function	
1	22c	From Edge Connector	28	5 Volts	
2	22a		27	CE_B	
3	21c		26	CE_A	
4	20c		25	$R/\overline{W}$	
5	20a		24	$\overline{VA}$	
6	19c		23	'Softy' A0	
7	18c		22	$\overline{EXTACC}$	
8	17c		21	19a	From Edge Connector
9	16c		20	18a	
10	15c		19	17a	
11	14c		18	16a	
12	13c		17	15a	
13	5a (A0 of Ext. MPU)		16	14a	
14	0 Volts		15	13a	

Edge Connector Signals are those from the Area Reserved for Control Signals, but are Otherwise Undefined.

Fig 6.13. 'Softy' Personality Socket Pinout.

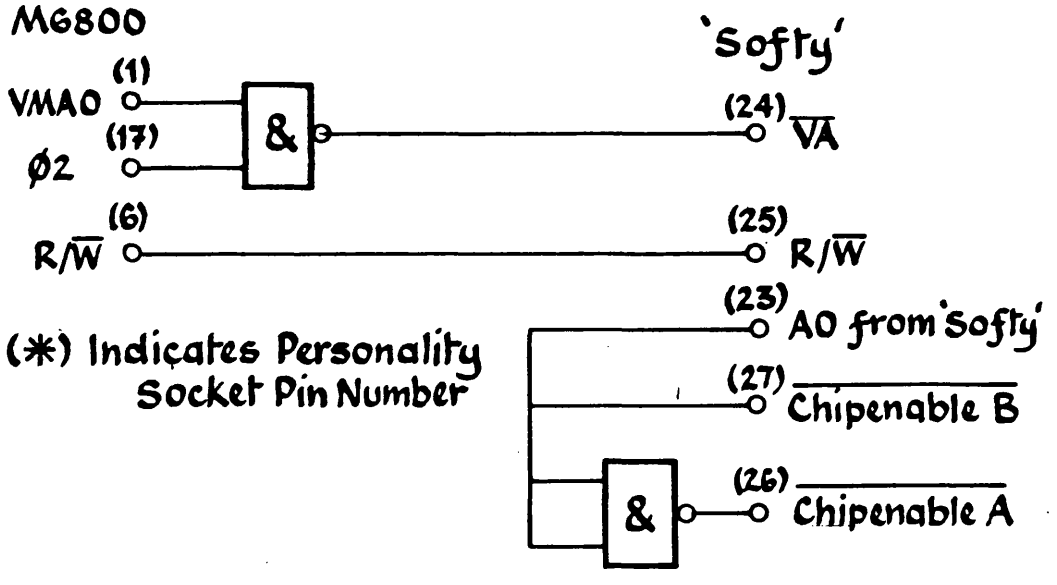


Fig. 6.14. M6800–'Softy' Personality Socket Circuit.

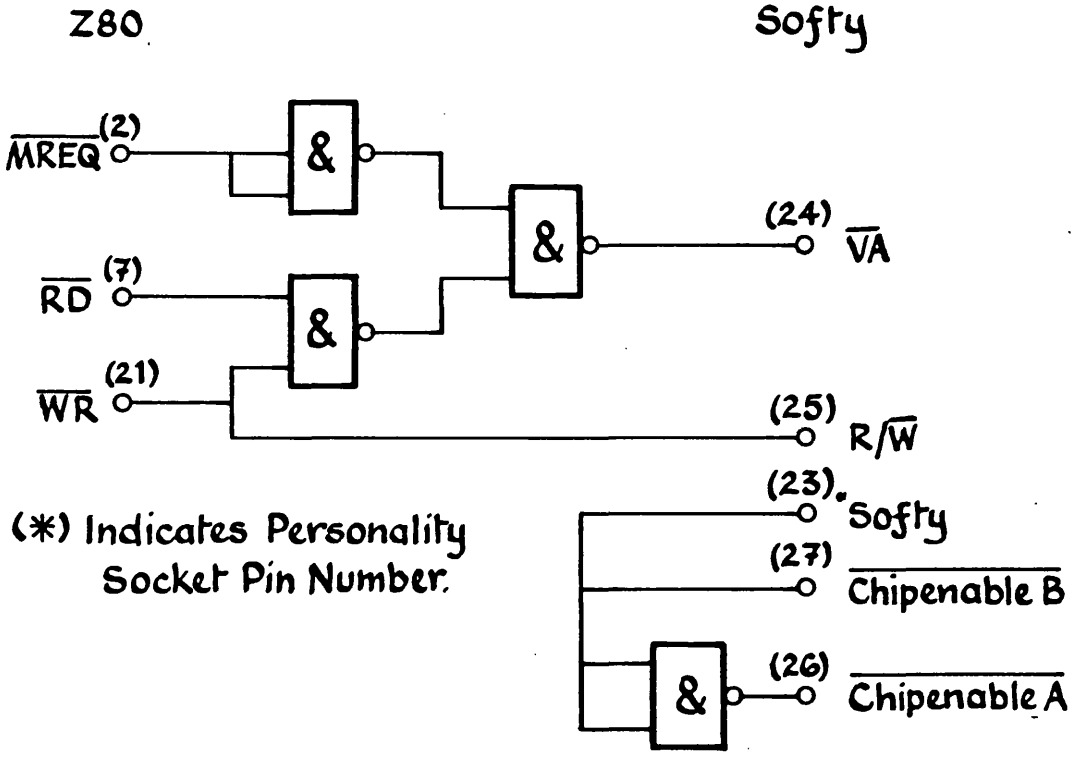


Fig 6.15 Z80-"Softy" Personality Socket Circuit

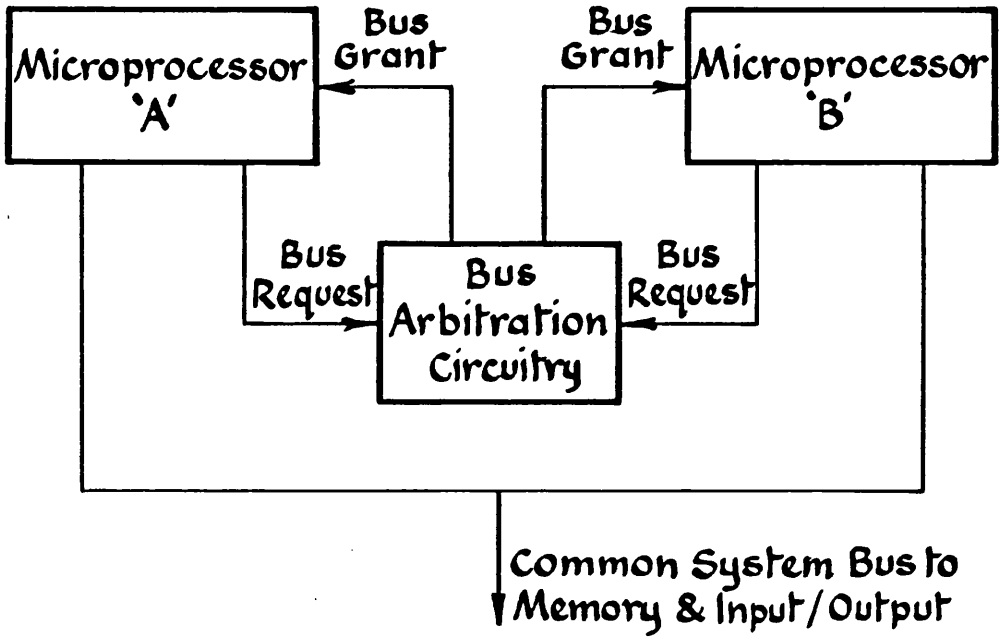


Fig 7.1. Connection of Two Microprocessors to Allow Access to Common Memory and Peripherals.

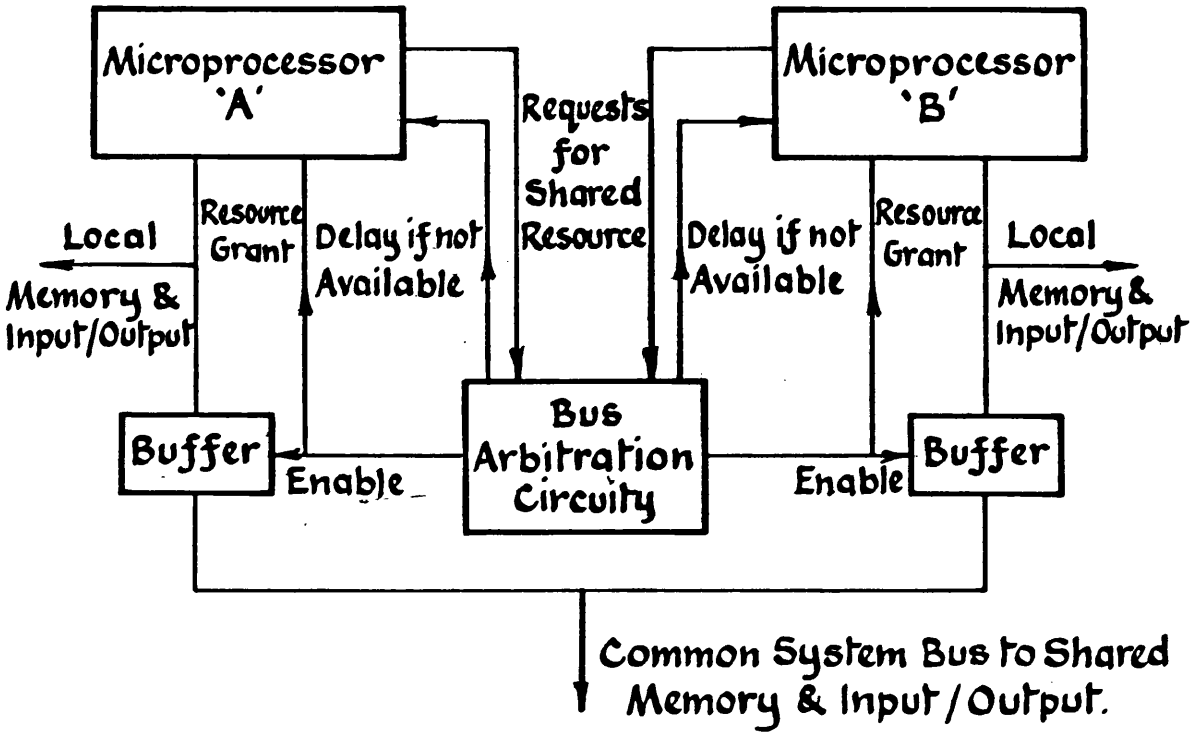


Fig. 7.2. Connecting Two Microprocessors (With Local Memory & Input/Output) to a Shared Resource.

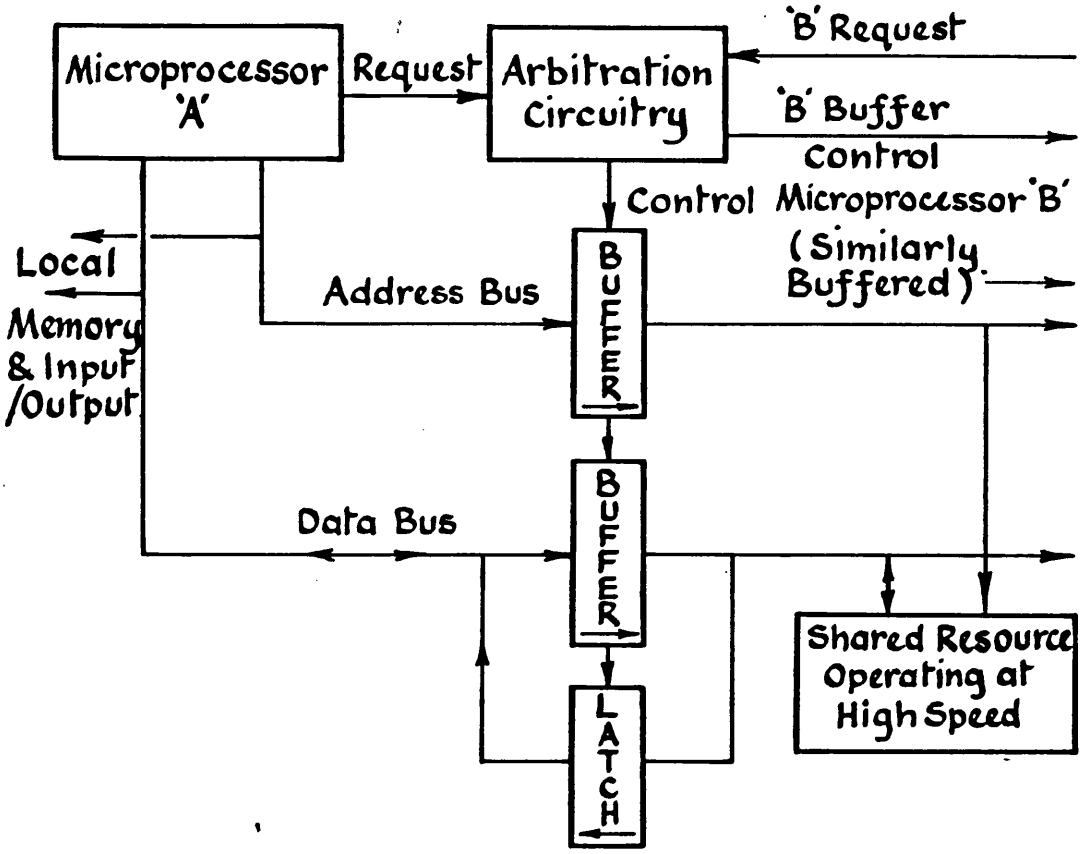


Fig. 7.3 Possible Connection of Two Microprocessors With Fixed Bus Cycle Time to Shared Resource.

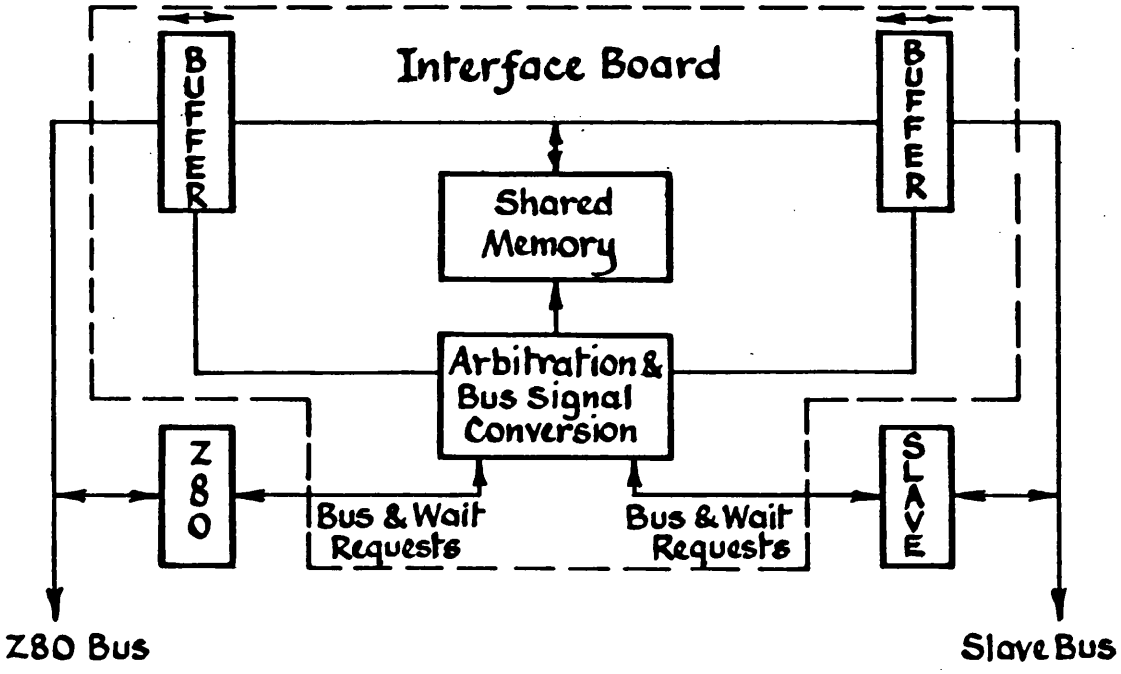


Fig. 7.4 Connection Scheme Adopted in this Investigation.

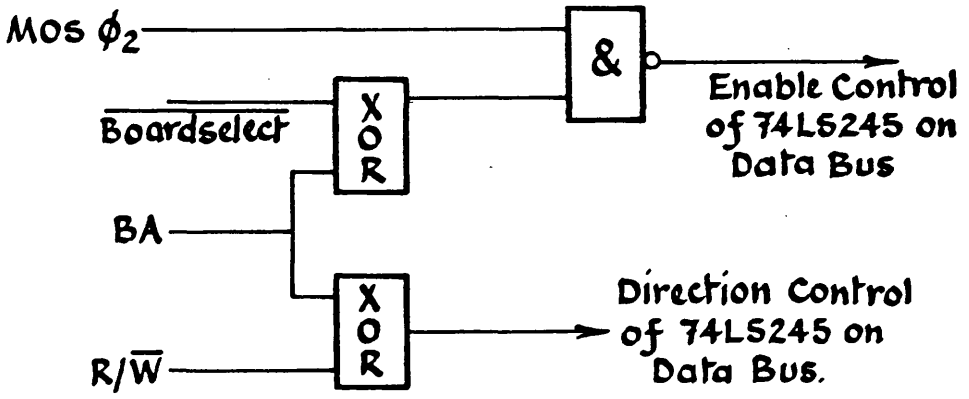


Fig. 8.1 Derivation of M6800 Data Bus Buffer Control Signals.

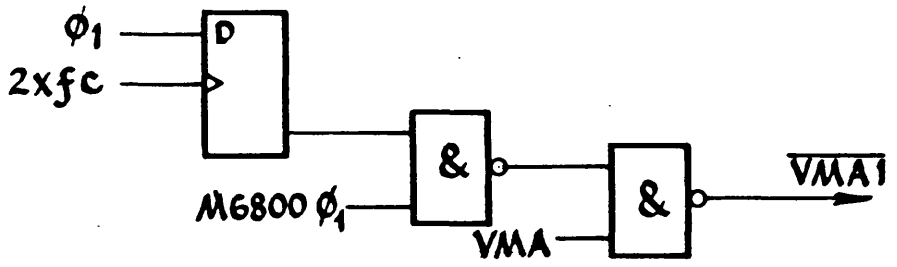


Fig 8.2 Generation of Modified VMA for Use on Interface Board

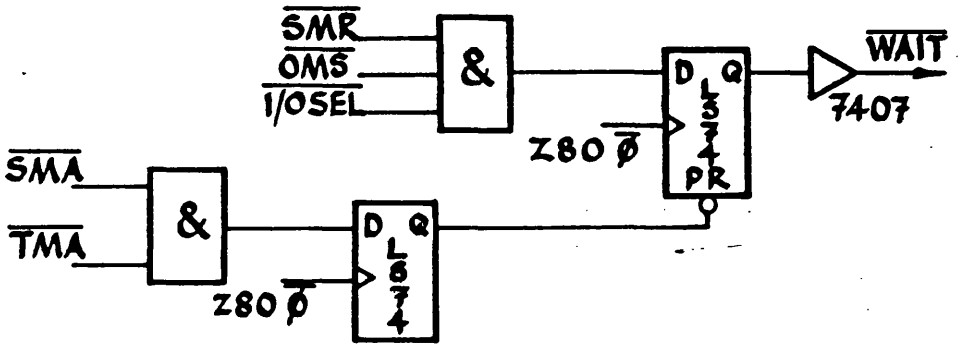


Fig 8.3 Z80 Wait State Generation Circuit

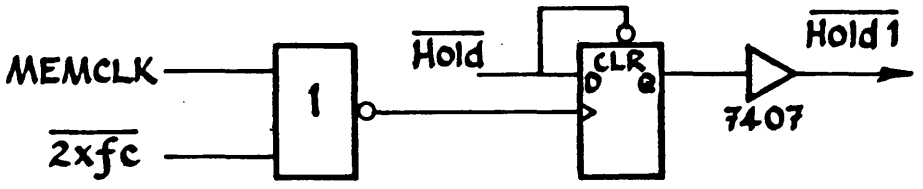


Fig 8.4 M6800 ϕ1 Hold Request Syncronisation

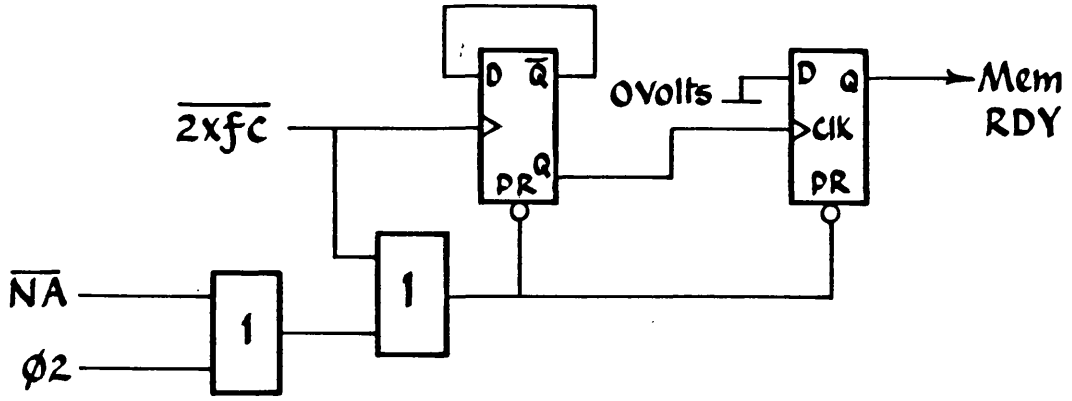


Fig. 8.5 Ø2 Stretching Circuit, M6800→Z80 Accesses

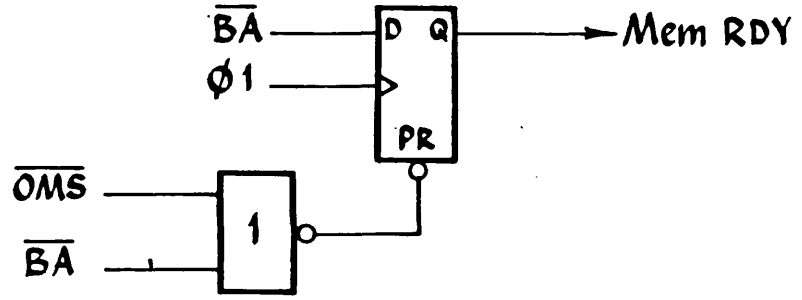


Fig 8.6 Ø2 Stretching Circuit, Z80→M6800 Accesses

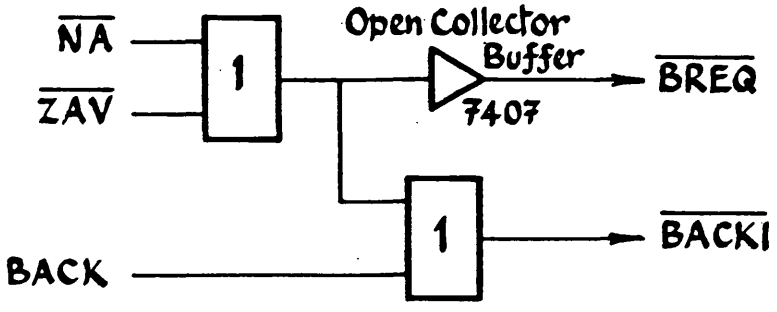
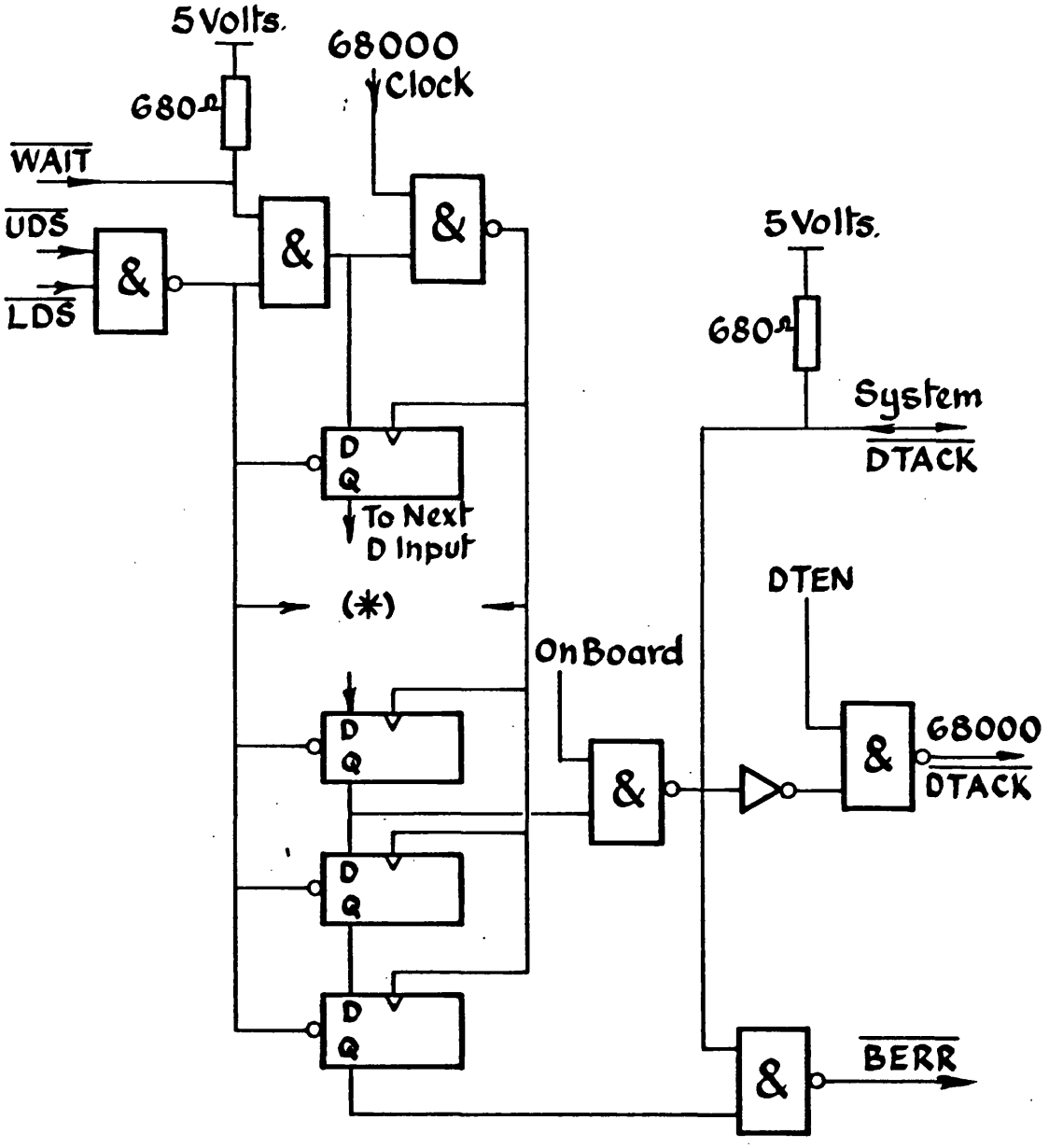


Fig 8.7 Generation of a $\overline{\text{BACK}}$ Signal that is Only Presented to the Interface After an M6800 Initiated Z80 Bus Request.



(*)The Number of Flip-Flops in the Chain Determines the Bus Cycle Length.

Fig. 8.8 MC68000 $\overline{\text{DTACK}}$ Generation Circuit

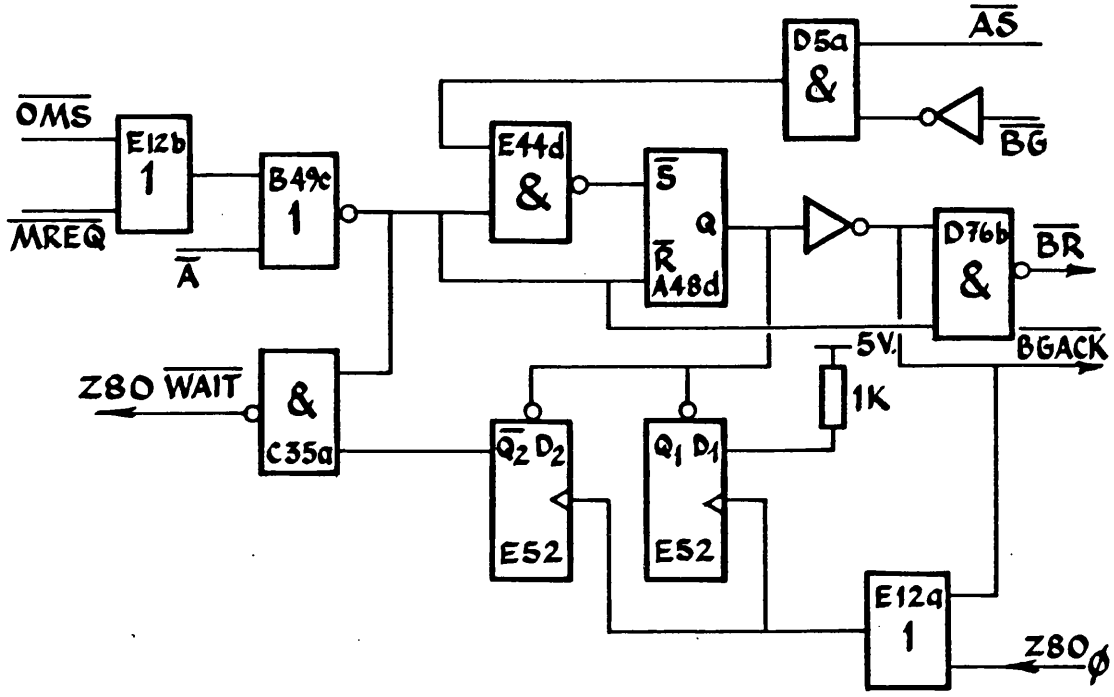


Fig 8.9. M68000 BGACK & Z80 Wait Generation Circuit

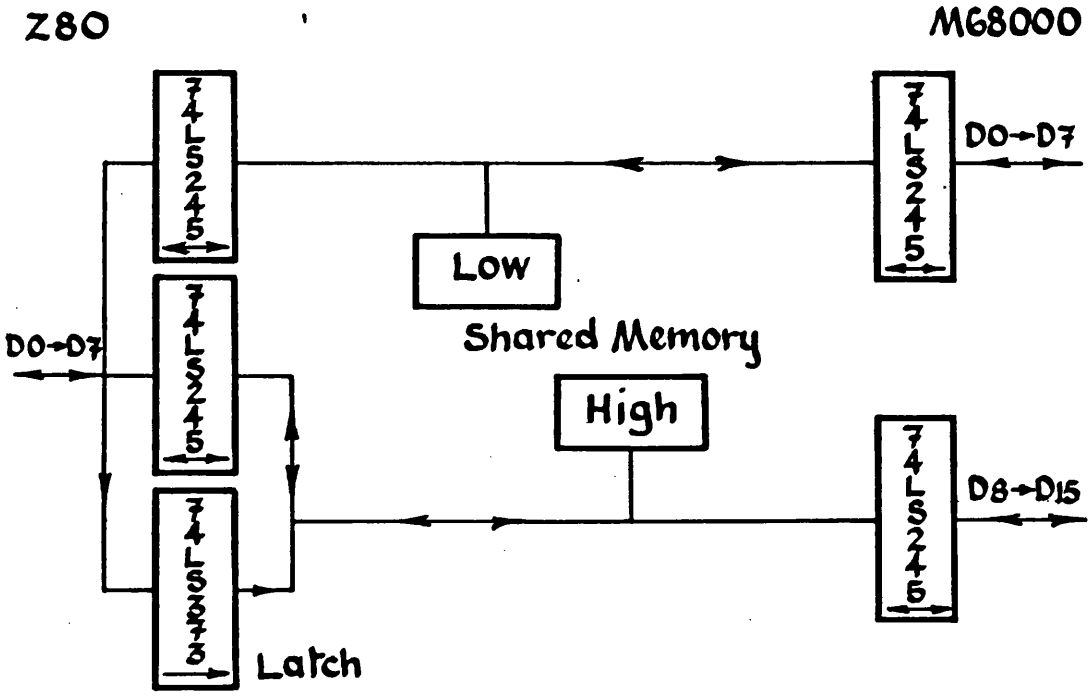


Fig. 8.10. M68000 Interface Board Data Paths

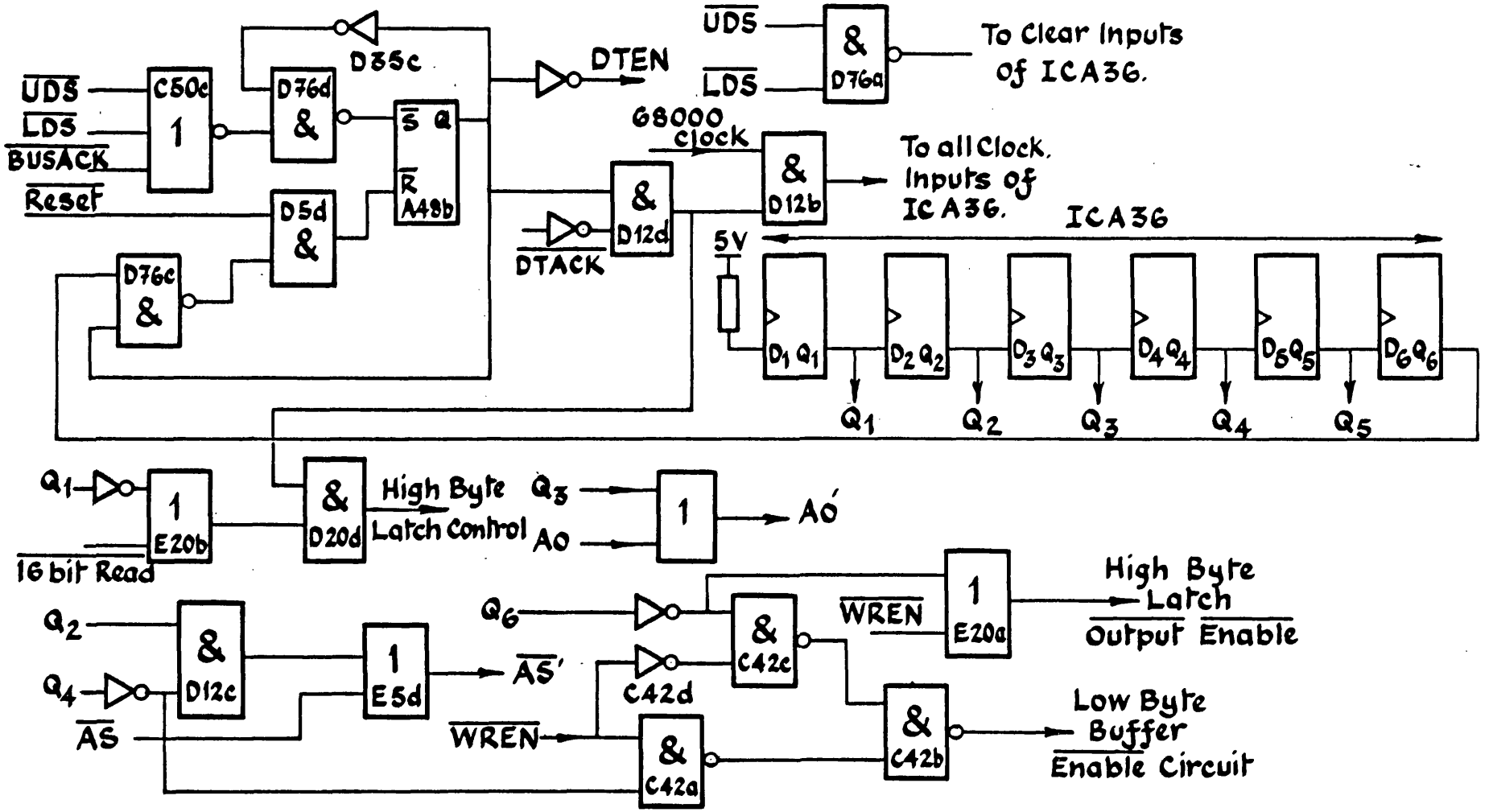


Fig 8.11 M68000 Interface, Z80 Word Access Control Circuit

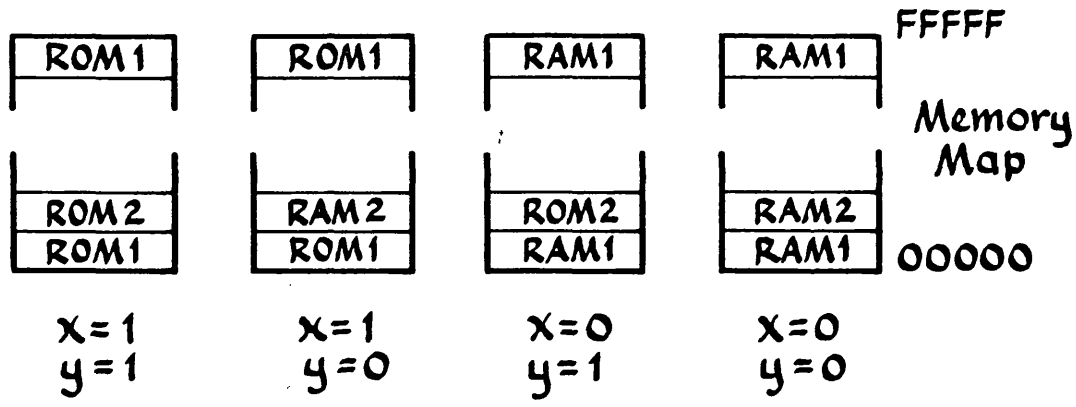


Fig. 8.12. Possible RAM/ROM Combinations at Intel 8086 Reset / Interrupt Locations.

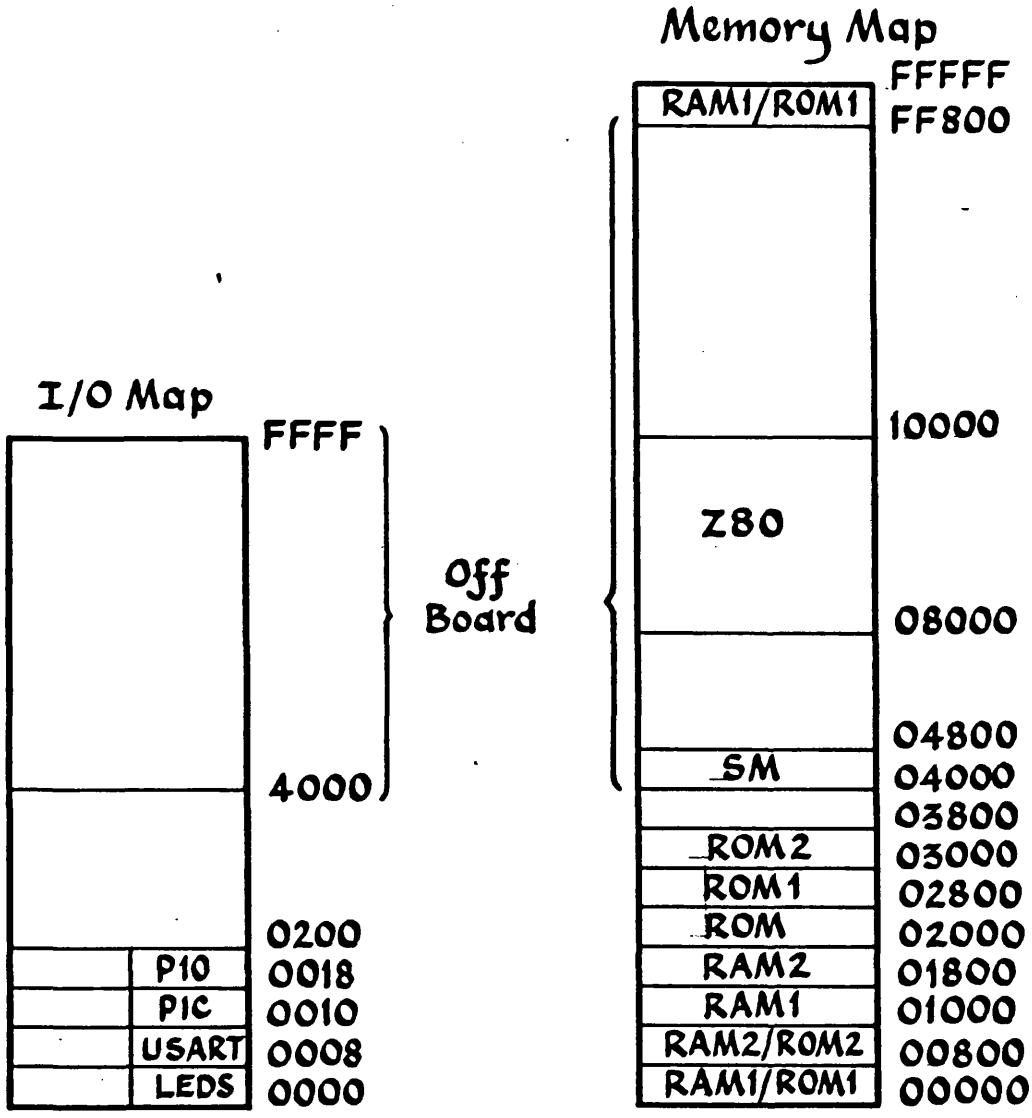


Fig. 8.13 Intel 8086 Memory & Input/Output Map.

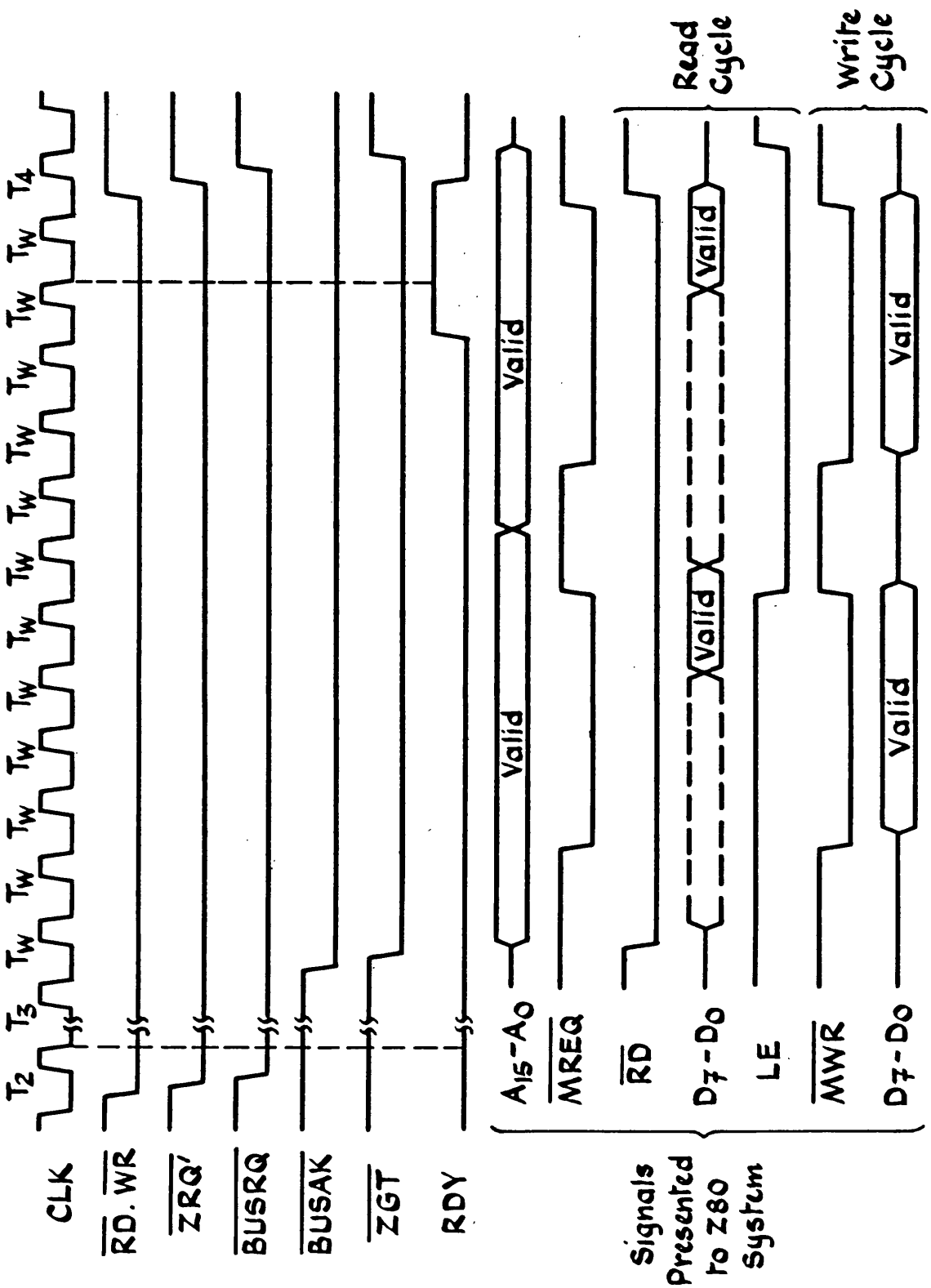


Fig. 8.14. Timing Diagram for 8086 Word DMA Into the Z80.

POLESTAR

ADVANCE PRODUCT INFORMATION
PRE-RELEASE

A NEW CONCEPT IN UNIVERSAL MICROPROCESSOR APPLICATIONS DEVELOPMENT AND EDUCATION

THE POLESTAR SYSTEM PROVIDES A LOW COST ENTRY INTO UNIVERSAL MICROPROCESSOR APPLICATIONS DEVELOPMENT WHICH IS CAPABLE OF CONTINUOUS EXPANSION TO A DISK BASED SYSTEM WITH FULL IN-CIRCUIT EMULATION FACILITIES SUPPORTING ALL THE POPULAR FAMILIES OF 8 BIT AND 16/32 BIT MICROPROCESSORS. EDUCATIONAL USERS CAN START WITH A LOW COST SINGLE BOARD STUDENT POSITION RUNNING OVER 30 K BYTES OF FIRMWARE WITH 8 K OF USER MEMORY PROVIDING SOPHISTICATED ASSEMBLY PROGRAMMING FACILITIES. MULTIPLE STATIONS OF THIS TYPE CAN BE LINKED VIA OUR POLESTAR LOCAL AREA NETWORK TO A CENTRAL DISK BASED HOST, ALLOWING SHARING OF DISK AND PRINTING FACILITIES BY EACH STUDENT.

SIRIUS MICROTECH LTD

Spring Lane South, Malvern Link, Worcs. Telephone: Malvern (06845) 66973

SHORT SPECIFICATION

POLESTAR SYSTEM

- Z80A host with 64 K bytes.
- System UART RS232 User UART RS232.
  System PIO User PIO.
- Target processor interface for 8/16 bit processors with high speed buffer.
- 600/1200 baud cassette interface.
- EPROM read/burn socket on front panel under software control for 2708, 2716, 2532, 2732, 2764.
- Power supply meeting most expansion requirements including floppy disk drives.

POLESTAR OPTIONS

- 128 K main memory.
- Low cost intelligent VDU controller for attached keyboard/monitor.
- Dynamically relocatable memory display of 512 bytes on separate monitor providing display of any 512 byte memory area or the last 512 bytes of I/O under software control.
- Floppy disk system – two double density 8 inch drives with controller.
- Polestar local area network.
- VDU's, printers, etc.
- Educational board with buffered PIO for relays, lamps, etc. plus A/D and D/A capability.

TARGET PROCESSORS

- Target chip set sockets for 8 × 2732 plus up to 8 K RAM, SIO (RS232), PIO.
- Polestar target processors – with modular personality firmware interfacing with system firmware and monitor to provide editor/assembler personalised to target.
  Supporting 68000, Z8000, 8086–6800, 8085, Z80, 6502 families (others to follow).
- Polestar in-circuit emulation – basic module plus personality modules for each processor supported providing probe/trace/event/trigger facilities with conditional operators and nested conditional statements under software control of disk based system.
- Student position can consist of any supported target processor running under the full Polestar firmware with 8 K of user memory and a CRTC handling a keyboard and screen. A power supply and 600/1200 baud cassette interface is provided to give a stand alone capability. A local area network interface is optional.

SOFTWARE/FIRMWARE

- Over 30 K of firmware provides a sophisticated development capability without disks.
  6 K monitor – editor – assembler – debug – disassembler – basic interpreter – test routines (diagnostics). Personalised to target by modules on target board.
- Software designed to run under *CP/M 2.2 DOS providing – monitor – editor – assembler – debug – disassembler – linker plus diagnostics. Local area network handler providing students with controlled access to disk and printing facilities. In-circuit emulation package.
- High level language support – PASCAL (others to follow).

* Trade Mark of Digital Research Inc.

Since our policy is one of continuous improvement this specification may be subject to change without notice.

SIRIUS MICROTECH LTD
Spring Lane South, Malvern Link, Worcs. Telephone: Malvern (06845) 66973